UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO FINAL DE GRADO

Grado en Enginyeria Electrónica Industrial i Automática

DESARROLLO DE UNA APLICACIÓN CON ARDUINO PARA LA MONITORIZACIÓN Y REGISTRO DE LOS DATOS DE PRODUCCIÓN ENERGÉTICA DE LA INSTALACIÓN SOLAR FOTOVOLTAICA DE LA EEBE.

Memória y Anexos

Autor: Juan Carlos Ruiz Agüera
Director: Guillermo Velasco

Convocatória: Enero 2019

Resum

A aquest apartat es pretén donar una visió general del projecte a realitzar. Sense entrar en detalls técnics peró deixant clares les funcionalitats del mateix.

El projecte consisteix a desarollar una aplicació que sigui capaç de monitoritzar i analitzar el comportament d'una instal·lació fotovoltaica. Per això, comptem amb un analitzador de xarxa i una placa ARDUINO MEGA 2560 amb la qual s'analitzaràn i trataràn les dades rebudes.

L'analitzador de xarxa es conectarà entre el convertidor DC/AC de la instalació fotovoltaica i la xarxa eléctrica per obtenir les dades de producció de la instalació. L'analitzador té sortides RS-485 i USB. En el nostre cas es connectará a través de la sortida RS-485 amb la que compte. A la **Figura 1** podem veure el diagrama de fluxe de la instal·lació.

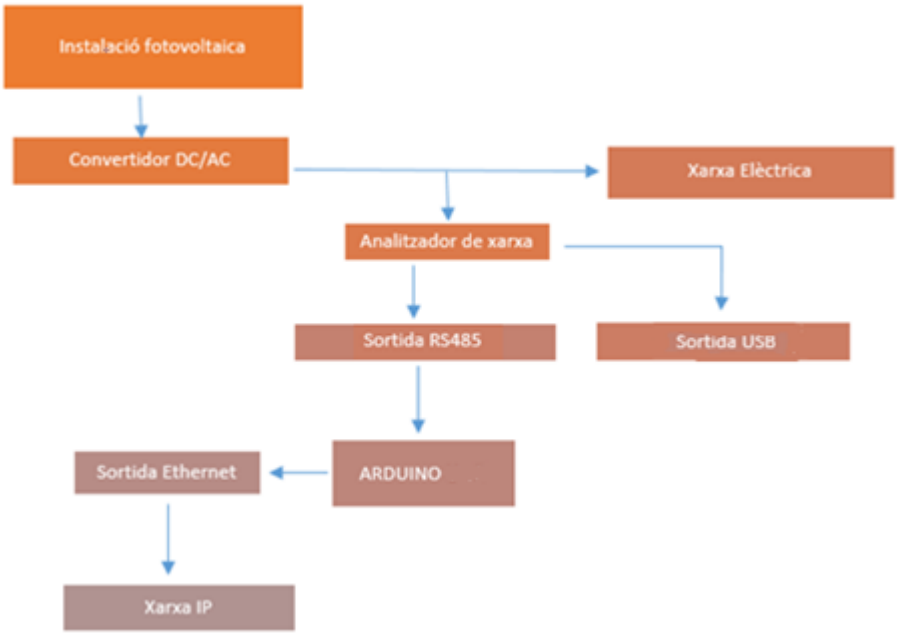


Figura 1. Diagrama de flujo de los datos (Fuente: Juan Carlos Ruiz).

Es programarà el software necessari pel tractament de les dades i es carregarà a la placa. D'aquesta manera, la placa ARDUINO Ethernet será capaç de tractar les dades d'entrada i enviar-les després a la xarxa per poder consultar-les a distancia, creant-se també un registre històric que podrà ser diari, setmanal, mensual... A més, es pretén també dotar el projecte amb un entorn gráfic per que en un futur les dades puguin ser mostrades a un monitor. Les dades monstrades serán tant de potències i energies generades per la instal·lació, com de la repercusió en el medi ambient que aquesta té.

Resumen

En este apartado se pretende dar una visión general del proyecto a realizar. Sin entrar en detalles técnicos pero dejando claras las funcionalidades del mismo.

El proyecto consiste en el desarrollo de una aplicación que sea capaz de monitorizar y analizar el comportamiento de una instalación fotovoltaica. Para ello, se cuenta con un analizador de red y una placa ARDUINO en la cual se analizarán y tratarán los datos recibidos.

El analizador de red se conectará entre el convertidor DC/AC de la instalación fotovoltaica y la red eléctrica para obtener los datos de producción de la instalación. Cuenta con salidas RS-485 y USB. En nuestro caso se conectará a la placa ARDUINO MEGA 2560 a través de la conexión RS-485 con la que cuenta. En la **Figura 2** podemos ver el diagrama de flujo de la instalación.

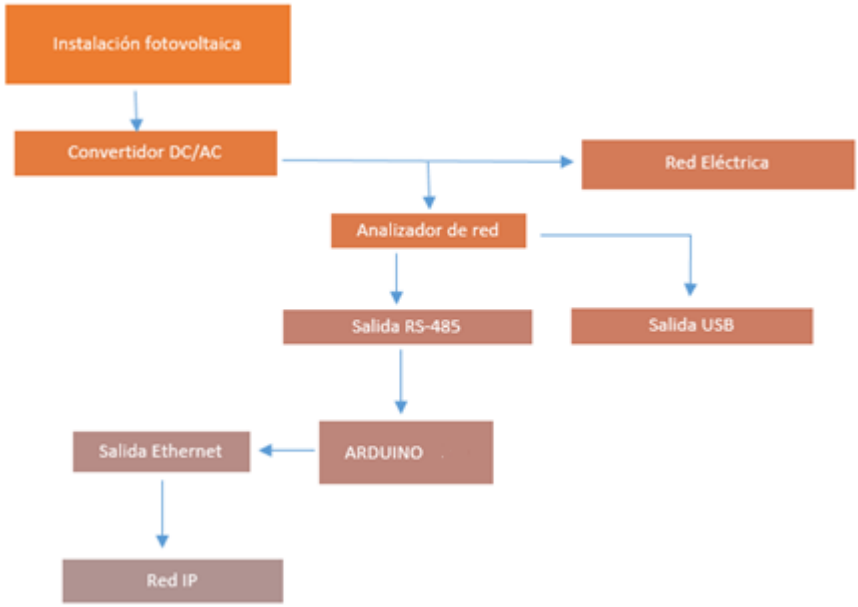


Figura 2. Diagrama de flujo de los datos (Fuente: Juan Carlos Ruiz).

Se programará el software necesario para el tratamiento de los datos y se cargará en la placa. De esta manera, la placa ARDUINO Ethernet será capaz de tratar los datos de entrada y mandarlos después a la red para poder consultarlos a distancia, creándose también un registro histórico que podrá ser diario, semanal, mensual… Además, se pretende también dotar al proyecto con un entorno gráfico para que en un futuro los datos puedan ser mostrados en un monitor. Los datos mostrados serán tanto de potencias y energías generadas por la instalación, como de la repercusión medioambiental que esta tiene.

Abstract

In this paragraph we talk about the project in a general way. We don't explain technical details but we will explain clearly its functionality.

The project consists in the development of an application that is able of monitoring and analyzing the behavior of a photovoltaic system. For this, it has a network analyzer and an ARDUINO board in which the data received will be analyzed and processed.

The network analyzer will be connected between the DC/AC converter and the electricity grid to obtain the production data of the installation. Analyzer have RS-485 and USB outputs. In the **Figure 3** you can see the flowchart of the installation.

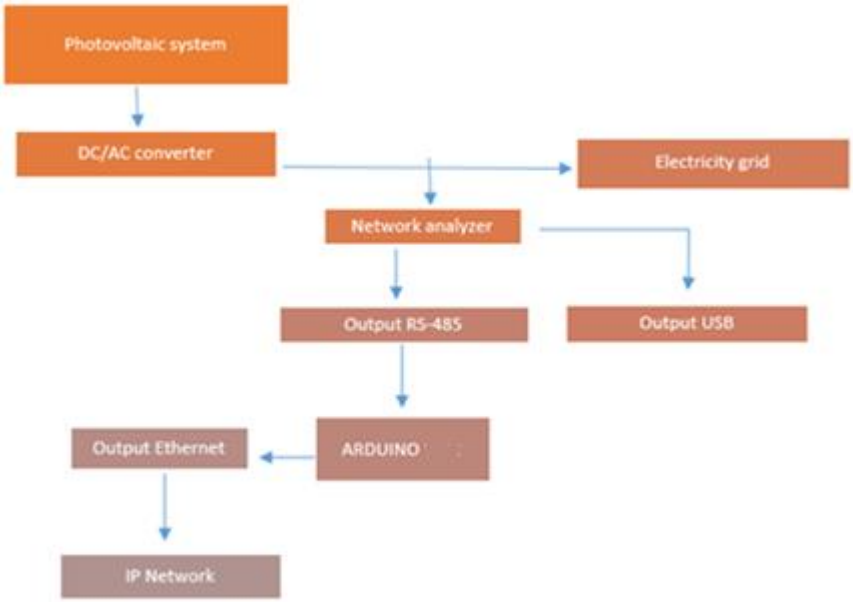


Figura 3. Diagrama de flujo de los datos (Fuente: Juan Carlos Ruiz).

The necessary software for processing the data will be programmed and loaded onto the board. In this way, the ARDUINO Ethernet board will be able to process the incoming data and then send them to the network to be able to consult them remotely.

In addition, an graphic environment will be implemented so that in future the data can be displayed on a monitor. The data shown will be the powers and energies generated by the installation. Also the environmental repercussion that this has.

Agradecimientos

Me gustaría dar las gracias a todos los que han colaborado en la realización de este proyecto de final de grado.

En primer lugar a mi tutor Guillermo Velasco, por su ayuda i compromiso durante la realización del proyecto.

También a mis compañeros, que me han dado su ayuda de manera totalmente desinteresada en todo momento, no solo durante la realización del proyecto sino también durante todo el grado. Habéis hecho de estos años que hemos estudiado juntos una gran experiencia. Tanto a los compañeros de la EUETIB como a anteriormente a mis compañeros de la EPSEM.

A mis amigos de toda la vida, gracias por darme apoyo en todo momento i por estar siempre a mi lado. Sois una pasada.

I finalmente, agradecer a mis padres por todos los esfuerzos que habéis hecho para que yo haya tenido la oportunidad de tener una buena educación. Espero haberla aprovechado como os merecéis.

Gracias por todo lo que habéis hecho, hacéis y haréis por mí. Solo puedo estar eternamente agradecido y orgulloso de teneros.

Desarrollo de una aplicación con Arduino para la monitorización y registro de los datos de producción energètica de la instalación solar fotovoltaica de la EEBE.

Glosario

- **Placa de desarrollo hardware:** es una computadora completa en un sólo circuito. El diseño se centra en un sólo microprocesador con la RAM, E/S y todas las demás características de un computador funcional en una sola tarjeta que suele ser de tamaño reducido, y que tiene todo lo que necesita en la placa base.

- **Arduino:** es una compañía open source y open hardware, así como un proyecto y comunidad internacional que diseña y manufactura placas de desarrollo de hardware.

- **Shield:** un shield en es una placa que se apila sobre el Arduino, de forma que nos permite ampliar el hardware o las capacidades del mismo.

- **Raspberry:** es un ordenador de placa reducida, ordenador de placa única u ordenador de placa simple (SBC) de bajo coste desarrollado en el Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de informática en las escuelas.

- **RS485:** es un estándar de comunicaciones en bus de la capa física. Está definido como un sistema de bus diferencial multipunto, es ideal para transmitir a altas velocidades sobre largas distancias (10 Mbit/s hasta 12 metros y 100 kbit/s en 1200 metros) y a través de canales ruidosos, ya que el par trenzado reduce los ruidos electromagnéticos que se inducen en la línea de transmisión.

- **Cable de par trenzado:** es un tipo de conexión que tiene dos conductores eléctricos aislados y entrelazados para anular las interferencias de fuentes externas y diafonía de los cables adyacentes.

- **Ethernet:** es un estándar de redes de área local.

- **Html:** es un lenguaje de programación que se utiliza para el desarrollo de páginas web.

- **Javascript (JS):** es un lenguaje de programación ligero, interpretado por la mayoría de los navegadores y que les proporciona a las páginas web, efectos y funciones complementarias a las consideradas como estándar HTML Este tipo de lenguaje de programación, con frecuencia son empleados en los sitios web, para realizar acciones en el lado del cliente, estando centrado en el código fuente de la página web.

- **Protocolo MODBUS:** es un protocolo de comunicaciones, basado en la arquitectura maestro/esclavo o cliente/servidor.

- **Analizador de red:** es un instrumento capaz de analizar las propiedades de las redes eléctricas.

Índice

1. Prefacio

En este apartado se pretende introducir el tema y las motivaciones del proyecto dentro del contexto medioambiental actual.

En estos tiempos de crisis energética en los que el precio del petróleo ha alcanzado máximos históricos, en los que se hace patente el uso excesivo de combustibles fósiles durante las últimas décadas, y en los que las ciudades están sufriendo más que nunca los efectos de la contaminación atmosférica que esto ha generado a causa de los gases tóxicos, se hace todavía más importante la búsqueda y utilización de alternativas para revertir esta situación.

El cambio climático es un tema que está en boca de todos. Gobiernos, empresas privadas y sector público están cada vez más concienciados y comprometidos con la conservación del medio ambiente y la reducción de las emisiones de gases contaminantes.

En este contexto, las energías renovables son la alternativa al uso de los combustibles fósiles. Por ello, entidades públicas y privadas invierten y apuestan cada vez más por investigación y adaptación de sus instalaciones para el uso de estas nuevas formas de energía mas limpias.

En el ámbito de las ciudades, cada vez es más común ver placas fotovoltaicas en los tejados de los edificios y construcciones. Dado el poco impacto visual y que no son necesarios grandes espacios para su instalación se han convertido en la primera fuente de energías renovables dentro de las ciudades. Además, la eficiencia de las placas fotovoltaicas se incrementa cada vez más, de forma que no hace falta contar con un gran espacio para tener un rendimiento más que considerable de la instalación.

Precisamente de la necesidad de monitorizar ese rendimiento de la instalación nace la idea del proyecto. La idea no es otra que la de recopilar toda la información posible de la instalación fotovoltaica a través de los datos proporcionados por un analizador de red y, a partir de esa información, buscar patrones de comportamiento de la instalación, picos de consumo, producción... Para de esta manera poder adaptarla de la mejor forma posible a nuestras necesidades.

En un principio el proyecto estaba planteado para aplicarlo sobre la instalación fotovoltaica de la Escola Universitaria d'Enginyeria Tècnica Industrial de Barcelona (EUETIB). Pero dada la reubicación sufrida por la escuela en el último año y que todavía no hay instalada ninguna instalación fotovoltaica en la EEBE (se espera que en un futuro si cuente con dicha instalación), se probará el proyecto con datos obtenidos de otra instalación pero se dejará preparado para su futuro uso en la EEBE.

La idea es que en un futuro los datos recopilados se podrán visualizar alrededor de la universidad en los monitores que hay repartidos por los pasillos, de esta manera los estudiantes serán más conscientes de el trabajo que esta haciendo el centro para favorecer el uso de energías renovables y se ayudará a esa concienciación sobre la necesidad de respetar el medio ambiente.

Además, los datos se podrán consultar a distancia, desde cualquier dispositivo conectado a la red del centro ya que los datos se mandarán a la red a través de la conexión Ethernet del dispositivo.

2. Introducción

El objetivo principal de este proyecto es la implantación de un dispositivo que sea capaz de monitorizar una instalación fotovoltaica a partir de los datos obtenidos por un analizador de red conectado entre el convertidor DC/AC del sistema y la red eléctrica.

Para ello se programará una placa ARDUINO con el software necesario para el tratamiento de los datos y se implementarán las comunicaciones con los diferentes dispositivos periféricos.

También se pretende generar un histórico con los diferentes datos obtenidos de manera que se puedan extraer conclusiones sobre su rendimiento a posteriori.

Otro de los objetivos del proyecto será displayar los datos tratados en un monitor a través de la red, en un entorno un poco más gráfico y ameno para el usuario. Ya que en un futuro se quiere implantar el sistema en la instalación fotovoltaica de la escuela y la idea es poder visualizar estos datos en monitores instalados en los pasillos de EEBE.

3. Ingeniería de la concepción

3.1. Estudio de las diferentes posibilidades

En este apartado se pretende realizar un estudio sobre las diferentes posibilidades y tecnologías que existen en el mercado actual para realizar un proyecto de las características del nuestro.

Para ello se hará una comparación entre dos de las herramientas más usadas en proyectos electrónicos: Raspberry Pi y Arduino. Existen varias diferencias entre estas dos herramientas, aunque la mayoría de proyectos se podrían realizar con ambas.

Arduino es un microcontrolador que podemos conectar a nuestro ordenador directamente a través de su bus USB y programarlo en función de las necesidades de nuestro proyecto, sin embargo no cuenta con un sistema operativo propio.

Figura 3.1. Arduino Uno (Fuente: www.arduino.cc).

Sin embargo es tanto hardware como software ya que gracias a los emuladores existentes podemos simular nuestros proyectos desde el ordenador personal. Podremos hacer conexiones virtuales y programarlo para comprobar cual sería su comportamiento real.

Arduino es por lo tanto una plataforma simple y dedicada a montar sobre ella los componentes necesarios para crear el proyecto.

Uno de los puntos fuertes del Arduino es su facilidad para conectarse con otros dispositivos a través de sus puertos de entrada/salida analógicas y digitales, programándolas fácilmente desde el software.

Además, cuenta con gran cantidad de shields de expansión capaces de equiparlo con funciones como conectividad WI-FI, GPS, conexión Ethernet, displays... Gracias a eso, Arduino tiene una gran facilidad de prototipado en muchos proyectos distintos.

Existen dos grandes tipos de proyectos en los que Arduino puede ser muy útil. Proyectos en los que se utiliza el microcontrolador de Arduino como único elemento inteligente. Y proyectos en los que Arduino se usa como interfaz.

Raspberry Pi, sin embargo, es un ordenador asequible. Cuenta con un microprocesador de entre 256MB y 1GB de memoria RAM dependiendo del modelo.

Raspberry tiene instalado un sistema operativo de LINUX. Cuenta con mucha más capacidad de cálculo y memoria, pero menos capacidad de conexión con periféricos y menos adaptabilidad en proyectos sencillos.

Figura 3.2. Raspberry Pi 3 (Fuente: www.raspberrypi.org).

La versión Pi modelo 2, por ejemplo, permite incluso reproducción de video en HD, tiene 4 puertos USB y cuenta con conexión a red Ethernet.

Normalmente Raspberry Pi suele ser utilizada más en proyectos informáticos que electrónicos.

En proyectos en los que la placa hace de interfaz Arduino suele adaptarse mejor, mientras que si nos interesa tener más capacidad de computación Raspberry Pi es la opción mas adecuada.

Pero no es necesario elegir siempre entre una u otra opción, ya que existen placas que mezclan características de las dos plataformas. Por ejemplo, la placa UDOO NEO. Aunque esta opción suele ser mas cara que las otras dos.

En cuanto al coste, las dos tienen precios muy similares. Las placas más básicas de ambas tienen un precio aproximado de entre 35 y 50 euros. Por lo tanto no es el precio una característica demasiado importante a la hora de elegir una u otra opción.

La diferencia por lo tanto la marca el hecho que Arduino se adapta mejor a operaciones de captura de datos y manejo de hardware externo.

3.2. Solución elegida

Como se ha comentado con anterioridad, el analizador de red dispone de dos posibilidades para la transmisión de los datos. Podemos utilizar un adaptador proporcionado por el fabricante (CIRCUITOR) de RS485 a USB y de esta manera recibir los datos a través del bus USB, o por el contrario, podemos recibir los datos directamente por el bus RS485 con el que cuenta el analizador.

Tanto Raspberry como Arduino necesitan de un componente (shield) externo para poder recibir los datos a través del bus RS485. Sin embargo, el manejo de estas placas de expansión és mas sencillo en Arduino.

Por lo tanto, dado el carácter electrónico de nuestro proyecto y la necesidad de utilizar diferentes componentes de hardware para la adquisición de los datos del analizador de red, se considera más eficiente para el desarrollo del mismo la elección de una placa Arduino.

A continuación se presentan los diferentes modelos de Arduino existentes actualmente. Primeramente hay que aclarar que existen dos tipos de productos de la marca Arduino en el mercado. Las placas que contienen el microcontrolador que controla el proceso y que se puede programar a través del bus USB. Y los shields que no son más que placas compatibles conectables a la placa principal y que la dotan de diferentes funcionalidades.

ARDUINO UNO

Se trata de la placa estándar, la más conocida y documentada. Se puso a la venta en septiembre de 2010 y contiene el microcontrolador Atmega328 con 32 Kbytes de memoria ROM para el código del programa. También cuenta con conexión USB HID.

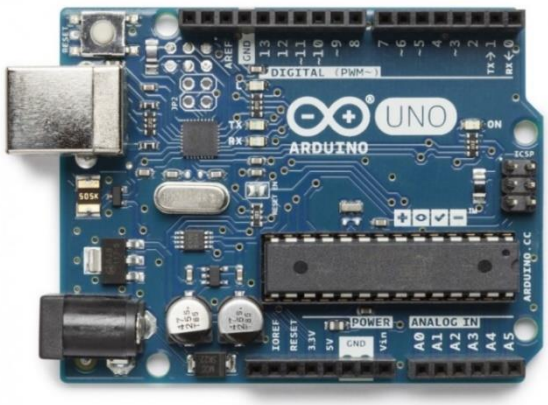


Figura 3.3. Arduino UNO (Fuente: www.arduino.cc).

ARDUINO MEGA

Este es el modelo más potente de Arduino. Es el que más pines de entrada/salida tiene. Está fabricado para realizar con él proyectos más complejos. Cuenta con un microcontrolador Atmega2560 con más memoria ROM para el programa y más memoria RAM que el resto de modelos.

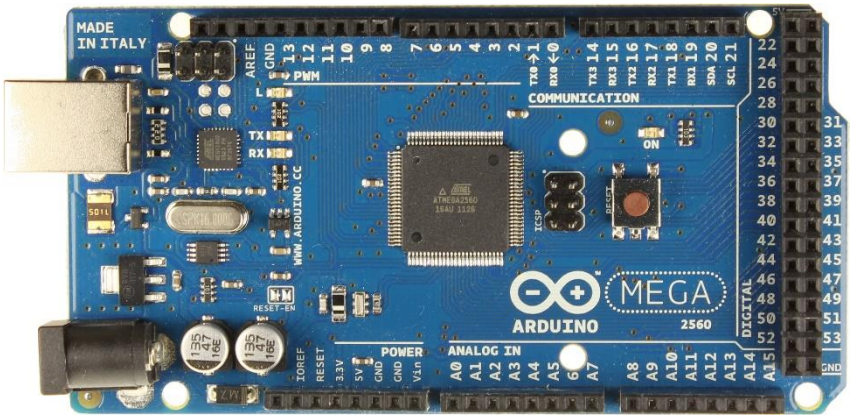


Figura 3.4. Arduino MEGA (Fuente: www.arduino.cc).

ARDUINO ETHERNET

Es básicamente una versión del ARDUINO UNO con conexión de red Ethernet. Sin embargo, no cuenta con conector USB. También cuenta con ranura para targeta de memoria Micro SD.

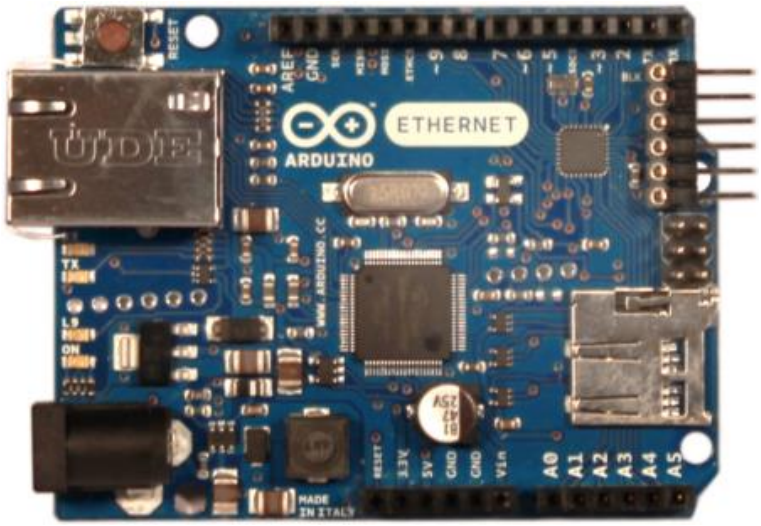


Figura 3.5. Arduino ETHERNET (Fuente: www.arduino.cc).

ARDUINO DUE

Este es el modelo con mayor capacidad de procesamiento. Contiene el microcontrolador Atmel SAM3X8E ARM Corte-M3 CPU. Se trata de un microcontrolador de 32 bit y arquitectura ARM. Otra gran diferencia de este modelo con el resto es que se alimenta a 3,3 V. Esto supone una limitación ya que la mayoría de shields, sensores y actuadores para ARDUINO se alimentan a 5 V.

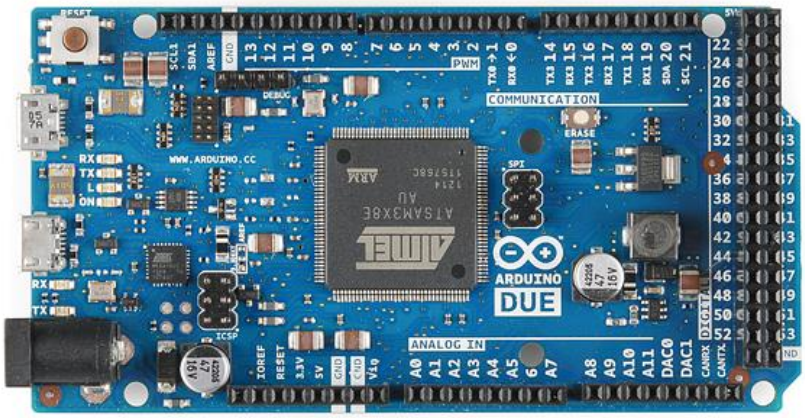


Figura 3.6. Arduino DUE (Fuente: www.arduino.cc).

ARDUINO LEONARDO

La diferencia de este modelo con el resto es que usa el microcontrolador ATmega32u4 que tiene integrado la comunicación USB, con lo cual se elimina la necesidad de usar un segundo microcontrolador para esta finalidad.

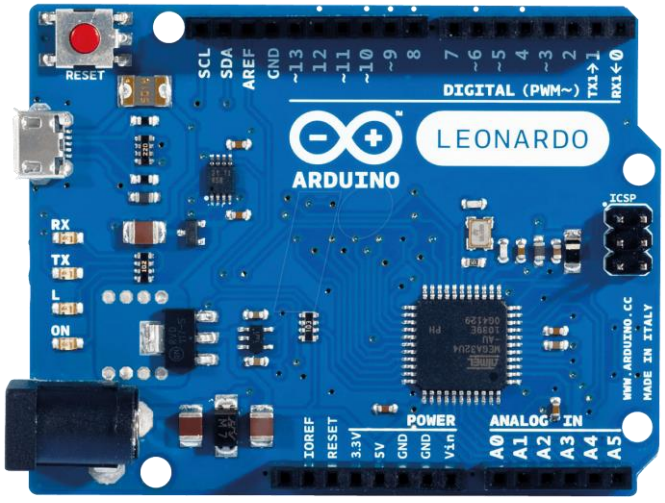


Figura 3.7. Arduino LEONARDO (Fuente: www.arduino.cc).

ARDUINO MICRO

Está basado en el ATmega32u4 del ARDUINO LEONARDO pero la placa tiene un formato mucho más compacto.

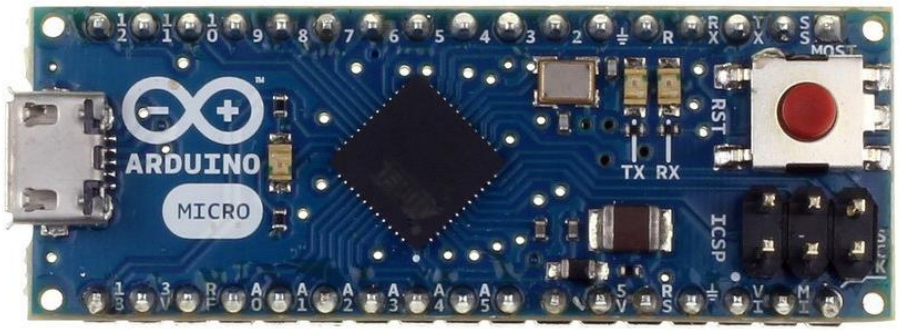


Figura 3.8. Arduino MICRO (Fuente: www.arduino.cc).

ARDUINO MINI

Es una versión más pequeña del modelo ARDUINO UNO. La única diferencia con este modelo es que no cuenta con conector USB.

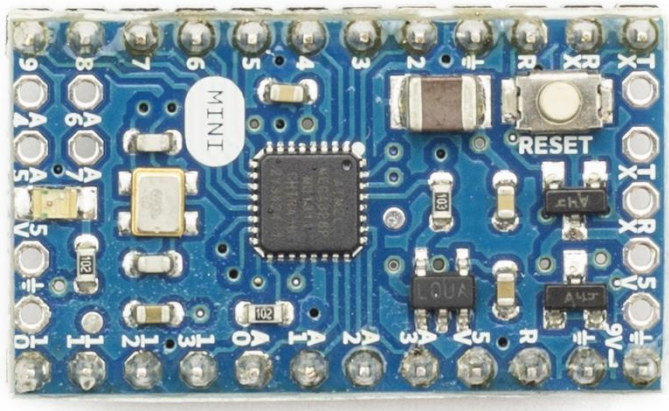


Figura 3.9. Arduino MINI (Fuente: www.arduino.cc).

Dadas las características del proyecto que vamos a realizar, se han valorado dos opciones de forma más detallada con la intención de elegir la opción que nos facilite posteriormente diseñarlo de la manera más cómoda.

La primera opción es utilizar la placa ARDUINO ETHERNET, teniendo en cuenta que esta no cuenta con conector USB.

La segunda opción es elegir el modelo ARDUINO MEGA junto con una placa shield de conexión a red Ethernet. Concretamente la conexión Ethernet se realizaría a través de la Ethernet Shield W5100.

Arduino Ethernet es una placa basada en el microcontrolador ATmega328, está basada en el modelo Arduino UNO. Dispone de 14 pines I/O, 6 entradas analógicas, un cristal de 16MHz, un conector de red RJ45, conector de alimentación, un zócalo ICSP para cargar el microcontrolador y un pulsador de RESET. Es la combinación en una sola placa de un Arduino UNO y una Ethernet Shield en un menor espacio físico.

En este modelo los pines 10, 11, 12 y 13 están reservados para la interfaz Ethernet y no se deben utilizar para otros propósitos. Esto reduce la cantidad de pines disponibles a 9 y a 4 con soporte PWM.

Una de las grandes diferencias de este modelo es que no dispone de un conector USB ni del chip conversor USB/Série, por lo tanto es necesario un cable FTDI 5V para poder programarlo lo que hace que sea una tarea mas árdua.

Dispone también de un zócalo para tarjetas de memoria MicroSD que puede ser utilizado para leer y escribir datos y es muy interesante en el caso de pequeños proyectos de servidores web. El pin 10 está reservado para la interfaz con el chip Wiznet. El pin SS para la tarjeta MicroSD está disponible en el pin 4. Esto hay que tenerlo en cuenta al utilizar las librerías que manejan la tarjeta SD.

El precio de esta placa es de 39,90 €.

Por otra parte, el modelo Arduino MEGA es una placa electrónica basada también en el microprocesador Atmega2560. Tiene 54 entradas/salidas digitales y 15 de estas pueden utilizarse para salidas PWM.

Además cuenta también con 16 entradas analógicas, un oscilador de 16MHz, una conexión USB al contrario que el modelo comentado anteriormente, un conector de alimentación, un header ICSP y un pulsador para el RESET.

Éste modelo incorpora autoselección del voltaje de alimentación (DC/USB) gracias a un chip MOSFET incluido en la placa. Además, dispone del bootloader OptiBoot que permite cargar programas a

115Kbps. Cuenta con salidas DC a 5V y 3,3V que hacen posible trabajar con prácticamente todo tipo de hardware externo.

A la placa, se conectaría la Ethernet Shield W5100. Está basada en el chip W5100 , con las librerías oficiales Ethernet, y permite la conexión del Arduino MEGA a una red Ethernet y su alimentación a través de esta línea.

El precio del Arduino MEGA es de 36,62€, mientras que el de la Ethernet Shield ronda los 10€. Por lo tanto el precio total de las dos estará alrededor de 45€.

De esta manera teniendo en cuenta las características del proyecto y que el espacio no es una cuestión critica en el proyecto, se opta por la segunda opción. De esta manera además, el ARDUINO se podrá cargar vía USB de manera más sencilla sin necesidad de usar un programador ICSP. Ahorrando también dinero ya que el precio de estas dos placas es menor que el del modelo ARDUINO ETHERNET.

El modelo Arduino MEGA tiene 256kb de memoria flash, 8 kb de SRAM y 4 kb de EEPROM.

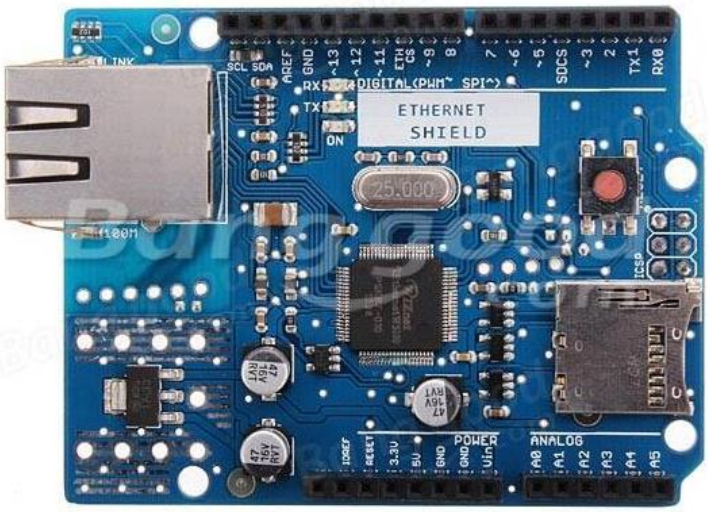


Figura 3.10. Arduino ETHERNET SHIELD (Fuente: www.arduino.cc).

En memoria flash es donde se guarda la imagen del programa y la data inicializada. Se trata de una memoria no volátil, no se borra cuando la alimentación se desconecta.

La memoria SRAM es la que se usa mientras el programa se está ejecutando. La usan las variables locales, las llamadas a funciones... Hay que tener cuidado con el uso que se le da a esta memoria ya que, por ejemplo, se requiere una gran cantidad de ella (512 bytes aprox.) para trabajar con las funciones propias del manejo de la tarjeta SD.

La memoria EEPROM es una memoria estática no volátil, más lenta que la memoria flash, a esta memoria solo podemos acceder byte a byte.

3.3. Funcionalidades

En este apartado se pretenden explicar con detalle las funcionalidades y el diseño del proyecto, haciendo énfasis en la comunicación entre el analizador de red y nuestro sistema de procesamiento de los datos basado en la placa ARDUINO MEGA 2560.

Para la obtención de los datos se utiliza un analizador de red CVM-SP-RS485-C de la marca CIRCUITOR. Este modelo cuenta con bus de comunicación RS-485 a través de la cual se transferirán los datos a la placa ARDUINO.

La gran ventaja de este tipo de comunicación es que se pueden realizar comunicaciones a largas distancias, incluso a 1200 metros. Además, se pueden transmitir los datos desde un solo dispositivo master a 32 dispositivos esclavos en paralelo.

En la escuela se dispone de un adaptador RS-485 a USB de forma que también se podría transmitir la información a través de este bus de comunicación si se requiere.

El analizador de red cuenta con un display en el que de manera secuencial se van mostrando todos los datos recogidos por el dispositivo. Este se conecta entre la salida del convertidor DC/AC de la instalación fotovoltaica y la carga del sistema eléctrico.

Figura 3.11. Analizador de red (Fuente: Juan Carlos Ruiz).

Los datos disponibles en el analizador son los siguientes:

- Voltaje [V].
- Corriente [A].
- Potencia Activa [kW].

- Potencia reactiva [kvarL/-C].
- Factor de potencia [PF].
- Cos α.
- Frecuencia [Hz].
- THD Voltaje [% THD-V].
- THD Corriente [% THD-A].
- Energia activa [W.h].
- Energia reactiva inductiva [kvarh.L].
- Energia reactiva capacitiva [kvarh.C].
- Potencia aparente [VA].
- Demanda máxima [Pd].

Se puede acceder a los datos parámetro a parámetro o recuperarlos todos a la vez. De esta manera, a través del bus RS-485 podemos consultar por una posición de memoria y longitud concreta o por todos los datos de los que dispone el analizador.

Para la comunicación se utiliza el protocolo estándar MODBUS que acepta baud rates de 1200, 2400, 4800, 9600 y 19200 baudios.

El formato de los mensajes para realizar consultas usando el protocolo de comunicación MODBUS-RTU (Remote Terminal Unit) es el siguiente:

- Código: 8 bits binarios con la información de la dirección del mensaje (el periférico al que consultamos).
- Función: función que vamos a realizar sobre el dispositivo.
- Bytes de información: paquetes de 8 bits con la información del mensaje.
- Chequeo del error: CRC (Cyclical Redunding Check). Se trata de la verificación del mensaje.

Así pues, mandando desde el Arduino a través del bus RS-485 la dirección del dispositivo al que queremos acceder (en nuestro caso el analizador de red), la acción que vamos a realizar y las posiciones de memoria de los datos que nos interesan y la longitud que estos tienen, el analizador nos devolverá la información solicitada.

Cabe destacar que el dispositivo Arduino MEGA no cuenta con un bus específico de protocolo MODBUS con lo cual se deberá utilizar una shield de expansión para dotarla de esta característica.

Existen cuatro funciones MODBUS disponibles para el analizador.

- 01: Lectura del estado del relé.
- 03 o 04: Lectura de n mensajes de 2 bytes de información de los parámetros eléctricos. La información de cada parámetro tiene una longitud de 32 bites por lo tanto son necesarios dos registros (4 bytes) para obtener toda la información del parámetro.
- 16: Escribir múltiples registros.

Para obtener los parámetros de comunicación usados por el dispositivo se debe mandar un mensaje con la estructura siguiente:

NP0403E80003CRC

La respuesta obtenida será del tipo:

NP0406aabbccddeeffCRC

Donde:

	Descripción	**Valor**		
NP	Número Periférico			
aa	Funcion Modbus	04		
bb	No. Periférico	01-FF (01-256 Hexadecimal)		
cc	Baud Rate	00 1200	02 4800	04 19200
		01 240	03 9600 (def.)	05 38400
dd	Bits de paridad	00 No	01 Par	02 Impar
ee	Bits de datos	8		
ff	Bits de stop	00 1 Stop bit	01 2 Stop Bit	

A continuación se muestra un ejemplo de mensaje para la adquisición de los datos:

0A 04 00 00 00 0A 71 76

Donde:

0A = Número de periférico del CVM-SP (10 en decimal)

04 = Función de lectura

00 00 = Dirección inicial de la lectura

00 0A = Número de registros a leer (10 en decimal)

7176 = Carácter CRC

La respuesta al mensaje seria la siguiente:

0A 04 14 00 00 08 4D 00 00 23 28 00 00 0F A0 00 00 00 90 00 00 00 60 CB 2E

Donde:

0A = Número de periférico del CVM-SP (10 en decimal)

04 = Función de lectura

14 = Bytes recibidos (20 en decimal, 2 por cada registro)

00 00 08 4D = V 1x10 (registro 00 Hex), en decimal 212,5 V

00 00 23 28 = mA 1, en decimal 9000 mA

00 00 0F A0 = W 1, en decimal 4000 W

00 00 00 90 = varL 1, en decimal 144 varL

00 00 00 60 = PF x 100 , en decimal 96

CB 2E = carácter CRC

Una vez tratados los datos recibidos, se mandarán a través de la comunicación Ethernet de la Shield W5100 a la red. De esta manera los datos formateados y tratados podrán ser consultados desde cualquier lugar a través de la red.

Esto dota al proyecto de mayor flexibilidad a la hora de recuperar y mostrar los datos, ya que estarán disponibles para cualquiera que se conecte a la red.

Puesto que la idea es que en un futuro estos datos se muestren en los monitores de la universidad, esto evita tener que realizar una comunicación vía RGB entre la placa Arduino y los monitores.

Los datos se mostrarán en un entorno web generado en lenguaje HTML y mandado a través del puerto ethernet.

La página tendrá un formato parecido al que se muestra a continuación en la **Figura 3.12**:

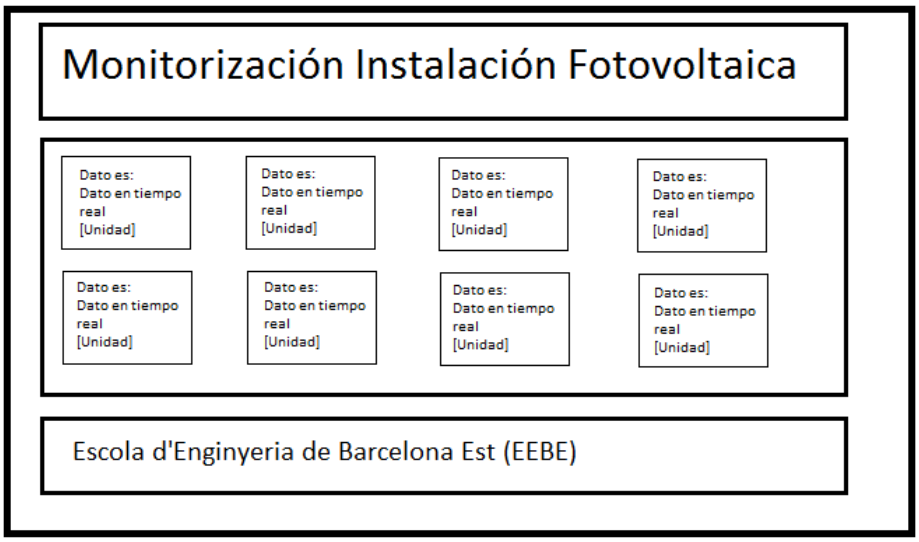


Figura 3.12. Formato página WEB (Fuente: Juan Carlos Ruiz).

Más adelante se detallará con más exactitud su aspecto.

3.4. Tratamiento de los datos

En este apartado se detalla el tratamiento que se dará a los datos una vez recibidos a través del bus RS485.

La consulta de datos se realizará cada 5 segundos. Se preguntará al dispositivo analizador de red por los 6 parámetros siguientes: voltaje, intensidad, potencia activa, factor de potencia, frecuencia y energía activa.

Una vez recibida la respuesta por parte del analizador, se convierten los datos a formato decimal y se registran en un fichero.

El mensaje de consulta es el siguiente: "0A04000000147EF1".

Donde:

Bytes	**Descripción**
01	Número de periférico del CVM-SP
04	Función de lectura
00 00	Dirección inicial de lectura
00 14	Número de registros a leer (20 en decimal)
7E F1	Caracteres CRC

Cada registro consta de 2 bytes, por lo tanto se leerán 40 bytes de los registros de información del analizador.

El mensaje de respuesta tiene el siguiente aspecto: "**0A 04 0c aa aa aa aa bb bb bb bb cc cc cc cc dd dd ee ee ff ff gg gg CRC"**

Se reciben 40 bytes en total con caracteres hexadecimales.

Donde:

Bytes	**Descripción**
01	Número de periférico del CVM-SP
04	Función de lectura
28	Bytes recibidos (40 en decimal)
aa aa aa aa	Bytes de informacion (4 bytes para cada parámetro)
CRC	Caracteres CRC (2 bytes)

Las posiciones de los registros de cada dato se muestran en la siguiente tabla.

Parámetro	**Posición en el array (decimal)**
Voltaje	[01,02]
Intensidad	[03,04]
Potencia Activa	[05,06]
Potencia Reactiva	[07,08]
Factor de Potencia	[09,10]
Cos α	[11,12]
Frecuencia	[13,14]
%THD V	[15,16]
%THD I	[17,18]
Energia Activa	[19,20]
Energia Reactiva Inductiva	[21,22]

Energia Reactiva Capacitiva	[23,24]
Potencia Aparente	[25,26]
Demanda Máxima	[27,28]

Para conversión de los datos primeramente se guardan todos los bytes recibidos en un array.Después se extrae cada dato numérico de cuatro bytes del array, para cada uno de los datos recibidos, extrayendo de la cadena recibida los bytes de las posiciones correspondientes a cada parámetro y guardándolos en una variable tipo int para su mejor manejo en el resto del proceso.

Pese a que los datos se reciben cada 5 segundos, no se graba un registro cada vez que se reciben. Los datos recibidos se displayan en tiempo real en la web enviada a través de la red Ethernet, a la vez que se hace la media de los datos en lapsos de aproximadamente 3 minutos. La media de estos 3 minutos es la que se registra en el fichero de la tarjeta SD.

El fichero se montará dándole el formato .csv adecuado para posteriormente poder exportarlo a la herramienta Excel de Microsoft Office.

La primera fila contendrá, separados por un punto y coma, los enunciados de las columnas del fichero. Las siguientes filas contendrán todos los datos de los parámetros recogidos a través del bus, en formato decimal y separados también por punto y coma. El primer dato de la fila será siempre la fecha y hora en que se han recogido esos datos.

La recogida de la hora se hará a través de una consulta a través de la red y se grabara con formato "hh:mm".

Con el registro de estos datos lo que se pretende es conocer mejor el comportamiento de la instalación fotovoltaica. Pudiendo así extraer patrones de comportamiento como: picos de consumo y producción, horas en las que se produce menos energía, horarios de demanda máxima de energía…

Pudiendo así adaptar la instalación para que el aprovechamiento de la energía sea máximo.

4. Ingeniería del desarrollo

En este apartado se pretende explicar el funcionamiento del proyecto desde un punto de vista más técnico, así como las diferentes fases de su desarrollo y las diferentes herramientas utilizadas.

Para desarrollar y compilar el código se ha utilizado el software disponible en la web de Arduino. El software cuenta con las librerías necesarias para la programación de la placa.

Para realizar pruebas antes de cargar el programa generado en la placa Arduino se ha usado el software Fritzing como herramienta de simulación. El programa dispone de los elementos necesarios para simular el montaje y las conexiones entre las diferentes shields, y cargar el código fuente para evaluar posteriormente su comportamiento.

A continuación se muestra el montaje en Fritzing para la simulación del proyecto:

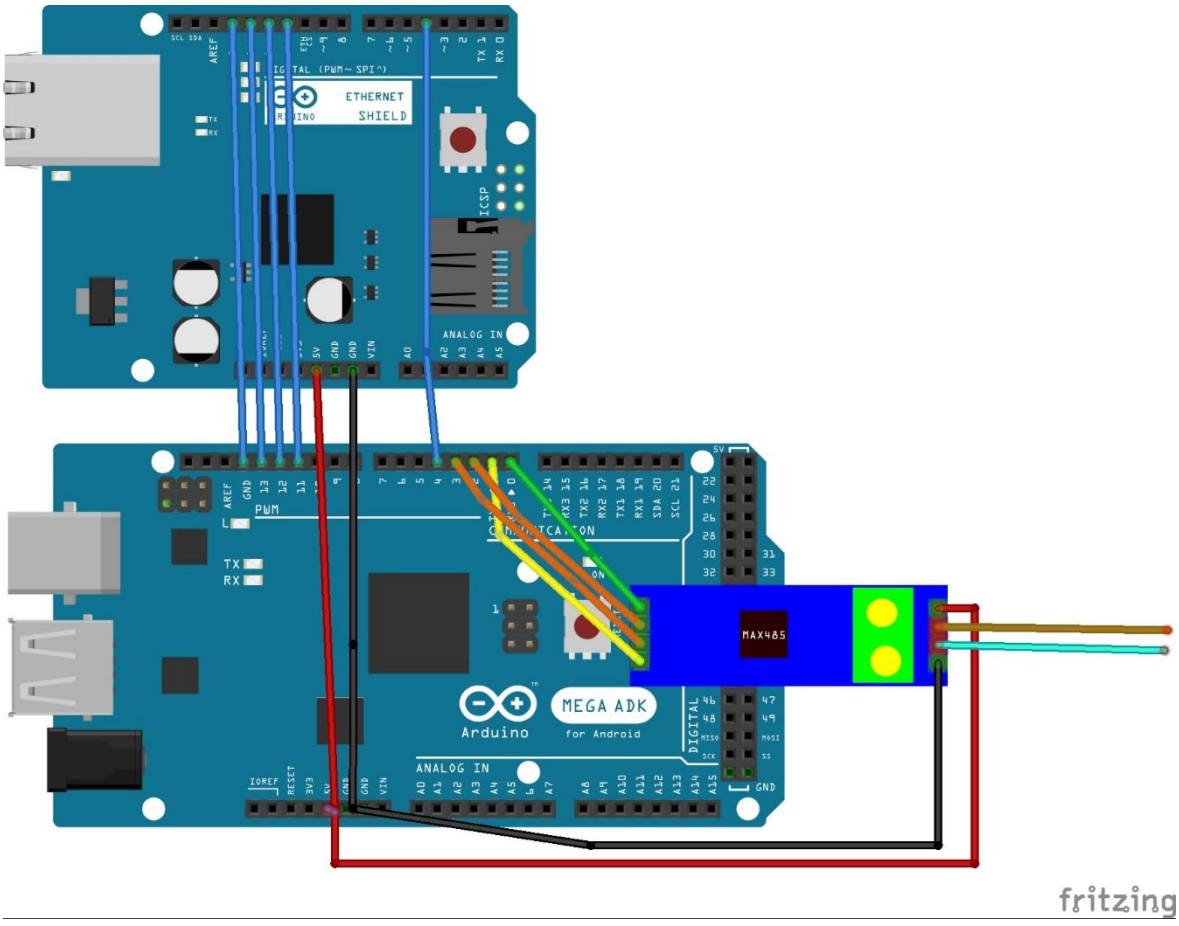


Figura 4.1. Esquema electrónico montaje (Fuente: Juan Carlos Ruiz).

Para conectar la placa de expansión Ethernet a nuestra placa Arduino MEGA no tenemos más que hacer coincidir los pines de las dos placas y ajustarlos hasta que queden bien conectados.

Como se observa en la **Figura 4.1**, los pines 4, 10, 11, 12 y 13 quedan utilizados en exclusiva por la Ethernet Shield.

Para conectar el driver de comunicación RS485 a la placa Arduino MEGA se conectan los pines de transmisión serie Rx y Tx a sus análogos en la placa Arduino (pines 0 y 1 respectivamente). Los pines R2 y R3 se conectan a dos de las salidas digitales de la placa, en nuestro caso los pines 2 y 3. De esta manera el dispositivo puede actuar en modo half duplex, tanto de receptor (poniendo la salida a 0) como de emisor (poniendo la salida a 1).

El aspecto del montaje se muestra en las **Figuras 4.2, 4.3 y 4.4**:

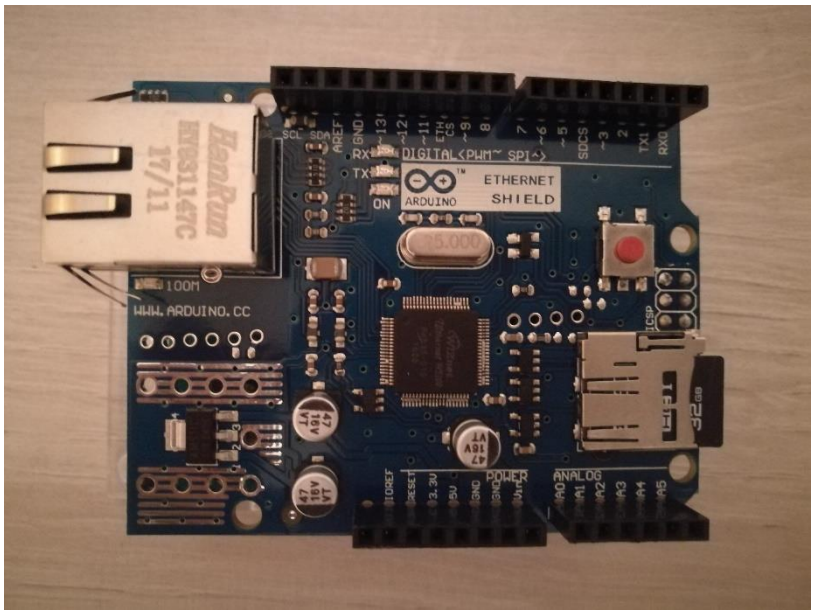


Figura 4.2. Ethernet Shield Vertical (Fuente: Juan Carlos Ruiz).

Vemos que la shield Ethernet dispone de una tarjeta de memoria Micro SD de 32 GB. En esta memoria se grabará el histórico de datos recibidos a través del analizador de red.

El PIN 10 queda reservado para la selección de la ethernet shield y el PIN 4 para activar el uso de la tarjeta SD.

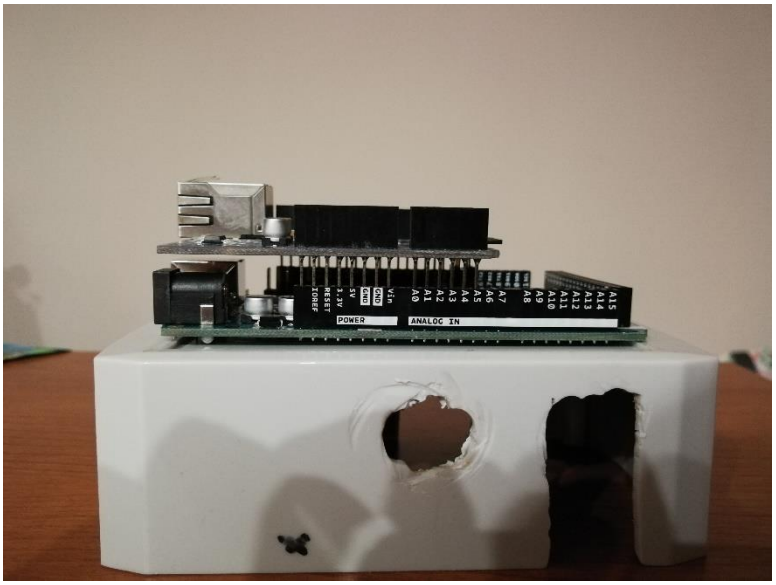


Figura 4.3. Encaje placas Lateral (Fuente: Juan Carlos Ruiz).

Los pines de la placa Arduino MEGA y Shield Ethernet quedan perfectamente alineados y encajados.

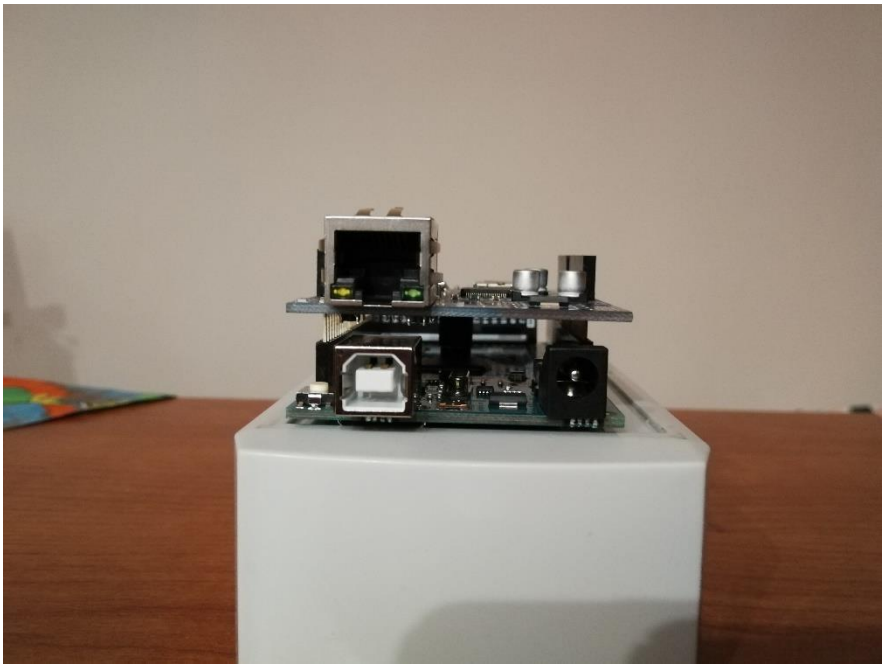


Figura 4.4. Encaje placas frontal (Fuente: Juan Carlos Ruiz).

Durante la ejecución del programa, la comunicación Ethernet y las funciones propias de la tarjeta SD no pueden estar activas al mismo tiempo.

El programa consta de varias partes bien diferenciadas. Tal como se muestra a continuacion:

```
INICIALIZACIÓN DE VARIABLES GLOBALES
SETUP {
        ESTABLECE COMUNICACIONES RS485 Y ETHERNET
        }
LOOP {
        RECUPERA FECHA Y HORA
        PREGUNTA AL ANALIZADOR
        ESPERA RESPUESTA
        TRATA DATOS RECIBIDOS
        CADA 3 MINUTOS {
                                ESTABLECE COMUNICACIÓN CON SD
                                GUARDA DATOS EN TARJETA SD
                                CIERRA COMUNICACIÓN SD
                        }
        SI HAY CLIENTE CONECTADO {
                                ESTABLECE CONEXIÓN CON SERVIDOR WEB
                                ENVIA WEB
                                CIERRA CONEXIÓN SERVIDOR WEB
                        }
```

En la primera parte definimos las librerías que van a ser usadas y las variables globales del programa.

En la función setup() se configuran las características de la comunicación serie y ethernet.

Dentro de la funcion loop() tenemos todo el procedimiento del programa, que se ejecutará de forma continua mientras la placa esté alimentada.

La función loop() tiene 4 partes bien diferenciadas.

En la primera parte se obtiene la fecha y hora desde la red.

Después se realiza la consulta al analizador de red. Lanzamos la pregunta a través del puerto serie y esperamos hasta que recibimos todo el mensaje de respuesta.

Una vez recibida la respuesta tratamos los datos recibidos, formateándolos de la manera más adecuada a las necesidades del proyecto.

Con los datos ya formateados, grabamos un registro en la tarjeta con todos los datos en decimal, separados por ";" para poder extraerlos en formato de hoja de cálculo. Esta parte se realiza únicamente una vez cada 3 minutos, grabando una media de los valores obtenidos durante ese periodo.

El registro empieza con la fecha y la hora en la que se han obtenido los datos.

Finalmente, en caso de haber algún cliente coonectado al servidor web, se mandan los datos a la red a través de la comunicación Ethernet.

A la hora de tratar la grabación del fichero en la tarjeta SD hay que tener varias consideraciones previas.

La placa Arduino MEGA 2560 solo es capaz de trabajar con formatos de datos FAT16 o FAT32, por lo tanto, tendremos que formatear la tarjeta previamente a empezar a trabajar con ella para asegurarnos que está configurada en uno de estos formatos.

Otra cosa a tener en cuenta es que la placa es capaz de leer y escribir en la tarjeta SD pero no es capaz de formatearla ni borrar datos, por lo tanto, debemos hacerlo previamente desde un ordenador.

Los nombres de los ficheros deben tener como máximo 8 caracteres y una extensión de 3 caracteres también.

También hay que tener cuidado a la hora de establecer conexiones ethernet y con la tarjeta SD ya que la shield Ethernet no puede trabajar con las dos a la vez.

El esquemático del montaje se muestra a continuación en la **Figura 4.5**:

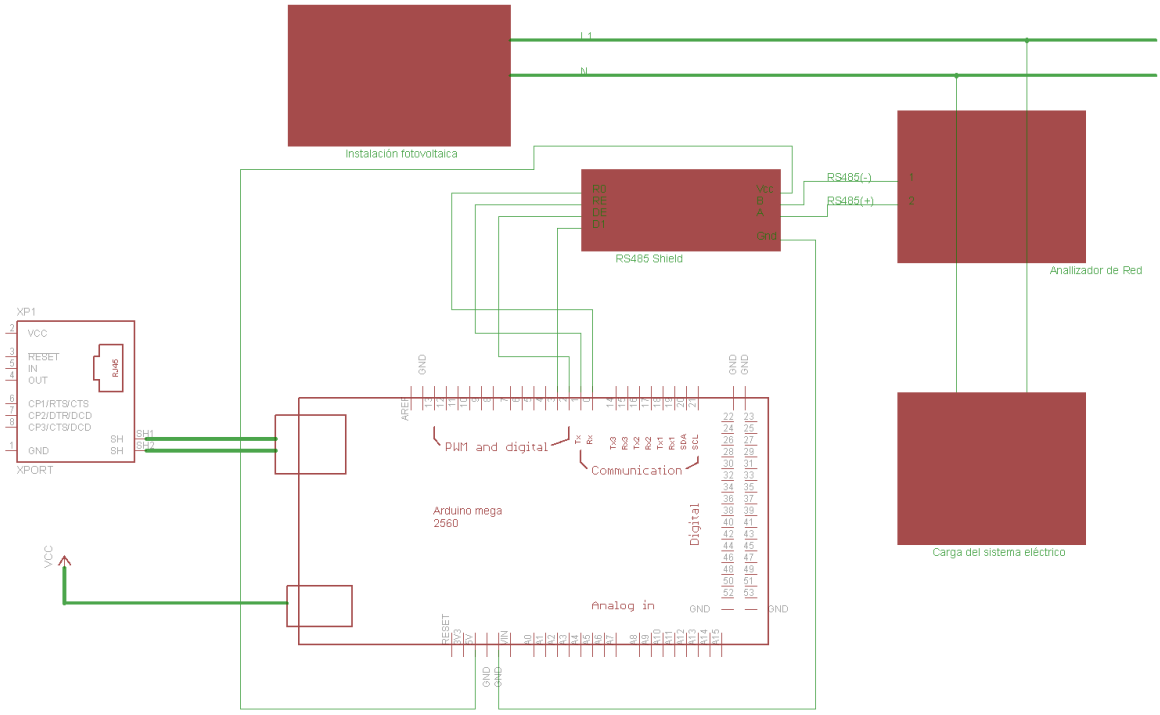


Figura 4.5. Esquema electrónico montaje (Fuente: Juan Carlos Ruiz).

Vemos como el shield de transmisión RS485 se controla a través de los pines Rx y Tx. La transmisión serie se ha configurado a una velocidad de 9600 baudios que es la velocidad estándar de la mayoría de los dispositivos. A las entradas A y B del MAX485 se conecta el analizador de red a través de su puerto de comunicación serie. La conexión entre la placa y la shield RS485 se realiza tal y como se muestra en la **Figura 4.6**:

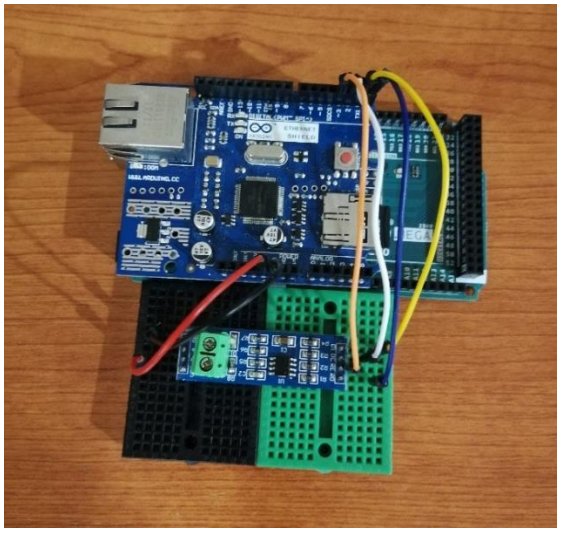


Figura 4.6. Montaje placa RS485 (Fuente: Juan Carlos Ruiz).

4.1. Módulo RS485

Para conectar el módulo MAX485 con la placa Arduino y el analizador de red, en primer lugar, se alimenta el módulo conectando Vcc y Gnd, respectivamente, a los pines 5V y Gnd de Arduino.

Por otro lado, se conectan los conductores A(RS485+) y B(RS485-) del par trenzado que constituyen el propio bus RS485, con los pines 2 (RS485+) y 1(RS485-) del analizador de red.

Ahora, se configura el módulo como emisor o receptor, para lo cual se emplean los pines RE (receiver enable) y DE (driver enamble). Si se conectan estos pines a Vcc el módulo actuará como emisor, y si se conectan a Gnd como receptor.

Finalmente, se conectan, respectivamente la entrada de datos al módulo DI (drive input) en el caso de que actué como emisor, o la salida de datos del módulo RO (receiver output) en el caso que actué de receptor.

Si queremos que durante la conexión el conversor RS485 pueda cambiar su papel de emisor a receptor (conexión half duplex) simplementese deben conectar los pines RE y DE a una salida digital para poder cambiar su tensión de Gnd a Vcc.

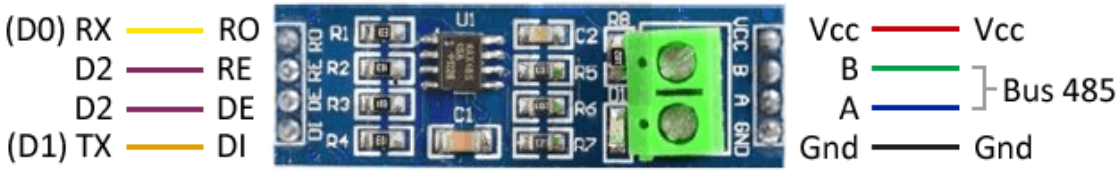


Figura 4.7. RS485 Shield (Fuente: https://aprendiendoarduino.wordpress.com).

El RS485 sólo es un protocolo de capa física, es decir, un "medio de transmisión" por el que se puede mandar cualquier señal digital, lo que incluye UART, Modbus, bus SPI, I2C.

Para conectarse con un dispositivo RS485, como por ejemplo un termostato, un sensor, o un actuador industrial, se necesita saber el protocolo que emplea y la trama a utilizar.

El siguiente ejemplo muestra el funcionamiento en modo half-duplex, es decir, cada dispositivo puede actuar como emisor o receptor, pero no simultáneamente. Para ello se dispone de dos pines digitales (pines 2 y 3 en nuestro caso) para cambiar el modo de cada dispositivo. También indicamos el modo en el que está operando cada dispositivo con el pin integrado en la placa.

El ejemplo simula la solicitud de un dato desde un Arduino master a un dispositivo slave. Para ello, el master envía un mensaje de solicitud. El slave responde enviando una string con los datos solicitados, que es recogido por el master. En un proyecto real podríamos enviar distintos comandos, o una dirección para comunicarnos con varios slaves.

```
const int RS485 =  2;  // HIGH = Driver / LOW = Receptor
void setup()
{
  Serial.begin(9600,SERIAL_8N1);
  Serial.setTimeout(100);

  pinMode(RS485, OUTPUT);
}
void loop()
{
  //RS485 transmisor
  digitalWrite(RS485, HIGH);
  Serial.print(mensaje_ida[i],BIN);
  Serial.flush();
  delay(50);

  //RS485 como receptor
```

```
digitalWrite(RS485, LOW);
if(Serial.available > 0)
{
  char data = Serial.readBytes();

  // Aqui hariamos lo que queramos con data
}
```

La configuración de la comunicación de nuestro analizador de red es la siguiente según su datasheet:

Numero de periférico = 1

1 = stop bits

N = parity

96 = 9600 baud rate

4.2. Protocolo Modbus RTU

Para implementar la comunicación entre la placa Arduino y el analizador de red se utiliza el protocolo de comunicación MODBUS. Existen dos formatos de comunicación Modbus, el RTU y el ASCII.

El analizador transmite los datos en formato binario, por lo tanto, usa el formato Modbus RTU. Este formato finaliza la trama con una suma de control de redundancia cíclica (CRC). En la **Figura 4.8** se comparan los dos protocolos Modbus.

	MODBUS/ASCII	**MODBUS/RTU**
Caracteres	ASCII 0..9 A..F	Binarios 0…255
Error Check	LRC	CRC
Frame inicio	:	3,5 chars de silencio
Frame final	CR/LF	3,5 chars de silencio
Gaps en el mensaje	1 seg	1,5 * longitud del char
Bit de start	1	1
Bits de datos	7	8
Paridad	Even/odd none	Even/odd none
Bit de stop	1 2	1 2

Figura 4.8. Protocolos comunicación MODBUS (Fuente: Juan Carlos Ruiz).

Nº Esclavo	**Codigo de operación**	**Subfunciones, Datos**	**CRC**

Figura 4.9. Trama MODBUS RTU (Fuente: Juan Carlos Ruiz).

La trama se transmite a través del puerto serie y sigue el orden que se muestra en la **Figura 4.10**.

START	DIRECCIÓN	FUNCIÓN	DATOS	CRC CHECK	STOP
T1-T2-T3-T4	8 BITS	8 BITS	N* 8 BITS	16 BITS	T1-T2-T3-T4

Figura 4.10. Trama MODBUS RTU con y sin bit de paridad (Fuente: Juan Carlos Ruiz).

Dirección (1byte): En el caso de las tramas enviadas por el máster, el campo de número de esclavo indica la dirección del destinatario de la trama. Permite direccionar hasta 247 esclavos, con las direcciones de 1 a 247 (0x00 a 0xF7). La dirección 0x00 es para los mensajes de Broadcast, así el primer esclavo comienza con la dirección 1 (de 1 a 247). En el caso de las tramas enviadas por los esclavos, este byte sirve para indicar al máster a quién pertenece la respuesta. Es decir, cada vez que un esclavo responde, sitúa su propia dirección en el byte de dirección lo que permite saber al maestro a que equipo corresponde cada respuesta. Las tramas broadcast, no tienen asociada respuesta, y algunas implementaciones de MODBUS no admiten la trama de broadcast.

Código de Operación o Función (1 byte): Indica el tipo de operación que queremos realizar sobre el esclavo. Las operaciones se pueden clasificar en dos tipos:

De lectura / escritura en memoria: para consultar o modificar el estado de los registros del mapa de memoria del esclavo.

Órdenes de control del esclavo: para realizar alguna actuación sobre el esclavo.

El código de operación puede tomar cualquier valor comprendido entre el 0 y el 127 (el bit de más peso se reserva para indicar error). Cada código se corresponde con una determinada operación. Algunos de estos códigos se consideran estándar y son aceptados e interpretados por igual por todos los dispositivos compatibles con MODBUS, mientras que otros códigos son implementaciones propias de cada fabricante. Es decir que algunos fabricantes realizan implementaciones propias de estos códigos “no estándar”.

Es también mediante el código de función que el esclavo confirma si la operación se ha ejecutado correctamente o no. Si ha ido bien responde con el mismo código de operación que se le ha enviado, mientras que si se ha producido algún error, responde también con el mismo código de operación pero con su bit de más peso a 1 (0x80) y un byte en el campo de datos indicando el código de error que ha tenido lugar. En la **Figura 4.11** se muestran las funciones MODBUS existentes en los dispositivos.

Función	Código Hexa	Tarea
0	00	Control de estaciones esclavas
1	01	Lectura de n bits de salida
2	02	Lectura de n bits de entradas
3	03	Lectura de n palabras de salida
4	04	Lectura de n palabras entrada
5	05	Escritura de un bit
6	06	Escritura de una palabra
7	07	Lectura rápida de 8 bits
8	08	Control de contadores de diagnósticos número 1 a 8
9	09	No utilizado
10	0A	No utilizado
11	0B	Control del contador de diagnósticos número 9
12	0C	No utilizado
13	0D	No utilizado
14	0E	No utilizado
15	0F	Escritura de n bits
16	10	Escritura de n palabras

Figura 4.11. Funciones MODBUS (Fuente: Juan Carlos Ruiz).

Dirección, datos y subfunciones (n bytes): Este campo contiene la información necesaria para realizar la operación indicada en el código de operación. Cada operación necesitará de unos parámetros u otros, por lo que el número de bytes de este campo variará según la operación a realizar. En el caso del esclavo, este puede responder con tramas con o sin campo de datos dependiendo de la operación. En los casos en que se produzca algún error es posible que el esclavo responda con un byte extra para especificar el código de error.

Cuando se produce un error en la ejecución de un comando en el esclavo, este responde poniendo a 1 el bit de más peso del código de función (0x80). Con este bit el maestro sabe que se ha producido un error, pero para obtener más detalle sobre el tipo de error, ha de comprobar el campo de datos. En la **Figura 4.12** se muestran los códigos de error posibles en la comunicación:

Código	Nombre	Significado
01	FUNCIÓN ILEGAL	El código de función recibido no se corresponde con ningún comando disponible en el esclavo.
02	DIRECCIÓN ILEGAL	La dirección indicada en la trama no se corresponde con ninguna dirección válida del esclavo.
03	DATO ILEGAL	El valor enviado al esclavo no es válido.
04	FALLO EN EL DISPOSITIVO ESCLAVO	El esclavo ha recibido la trama y la ha comenzado a procesar, pero se ha porducido algún error y no ha podido finalizar la tarea.
05	ACKNOWLEDGE	El esclavo informa al master que tardará en procesar la tarea. Así evita que el master considere un error de timeout. El master podrá enviar mas tarde una trama de tipo Poll Program Complete para verificar si se ha completado el comando.
06	ESCLAVO OCUPADO	El esclavo esta ocupado realizando otra tarea.

Figura 4.12. Códigos de error MODBUS (Fuente: Juan Carlos Ruiz).

Control de errores LRC o CRC: Se utiliza un sistema de detección de errores diferente dependiendo del tipo de codificación utilizado (ASCII o RTU). En el caso de la codificación ASCII es el checksum (o

Longitud Redundancy Check LRC) en módulo 16 expresado en ASCII (2 caracteres representan 1 byte), sin considerar el ":" ni el "CR LF" de la trama. En la codificación RTU se utiliza el método de CRC (Cyclical Redundancy Check) codificado en 2 bytes (16 bits).

Para calcular el CRC se carga un registro de 16 bits todo con '1's, se hace OR con cada uno de los caracteres de 8 bits con el contenido de cada byte y el resultado se desplaza un bit a la izquierda insertando un 0 en la posición de menor peso (la de la derecha). El de la izquierda se extrae y se examina: si es 1 se vuelve a hacer OR con un valor prefijado, si es 0 no se hace ningun OR… y el proceso se repite hasta que se han hecho los 8 shifts del byte.

Evidentemente, existen herramientas y aplicaciones que calculan el CRC directamente introduciendo la trama del mensaje que se va a enviar a través del protocolo MODBUS,por ejemplo:

https://www.lammertbies.nl/comm/info/crc-calculation.html

4.3. Librería Modbus para ARDUINO MEGA 2560

Para implementar el protocolo MODBUS en nuestro proyecto Arduino se utilizará la librería ModbusMaster.

Esta librería sirve para establecer comunicación con cualquier esclavo utilizando el protocolo Modbus RTU. Para ello se crea un paquete con toda la información necesaria para comunicarse con el esclavo, se envía el paquete y se espera para recibir la respuesta.

A continuación se explica con más detalle el funcionamiento de la librería y su implementación dentro de nuestro proyecto.

Dentro de la librería ModbusMaster.h encontramos definida la clase ModbusMaster junto con todas las funciones necesarias para establecer la comunicación. Las funciones utilizadas son las siguientes:

ModbusMaster

```
ModbusMaster::ModbusMaster(void)
{
  _idle = 0;
  _preTransmission = 0;
  _postTransmission = 0;
}
```

Se ejecuta cuando se crea el class object.

node.begin

```
void ModbusMaster::begin(uint8_t slave, Stream &serial)

{

  _u8MBSlave = slave;

  _serial = &serial;

  _u8TransmitBufferIndex = 0;

  u16TransmitBufferLength = 0;

#if __MODBUSMASTER_DEBUG__

  pinMode(__MODBUSMASTER_DEBUG_PIN_A__, OUTPUT);

  pinMode(__MODBUSMASTER_DEBUG_PIN_B__, OUTPUT);

#endif

}
```

Assigna la dirección del esclavo y el puerto serial de la comunicación.

node.preTransmission

```
void ModbusMaster::preTransmission(void (*preTransmission)())
{
  _preTransmission = preTransmission;
}
```

Esta función se llama justo antes de que un mensaje Modbus se envíe a través del puerto serie y habilita el transreceptor RS485.

node.postTransmission

```
void ModbusMaster::postTransmission(void (*postTransmission)())
{
  _postTransmission = postTransmission;
}
```

Esta función se llama justo después de que un mensaje Modbus se envíe a través del puerto serie y deshabilita el transreceptor RS485.

node.readInputRegisters

```
uint8_t ModbusMaster::readInputRegisters(uint16_t u16ReadAddress,
  uint8_t u16ReadQty)
{
  _u16ReadAddress = u16ReadAddress;
  _u16ReadQty = u16ReadQty;
  return ModbusMasterTransaction(ku8MBReadInputRegisters);
}
```

Esta función recibe las características de la lectura que se quiere realizar en los registros del esclavo y llama a la función ModbusMasterTransaction.

Devuelve el resultado de la transmisión y el motivo en caso de que no sea exitosa.

ModbusMasterTransaction

```
uint8_t ModbusMaster::ModbusMasterTransaction(uint8_t u8MBFunction)
{
  uint8_t u8ModbusADU[256];
  uint8_t u8ModbusADUSize = 0;
  uint8_t i, u8Qty;
  uint16_t u16CRC;
  uint32_t u32StartTime;
  uint8_t u8BytesLeft = 8;
  uint8_t u8MBStatus = ku8MBSuccess;

  // assemble Modbus Request Application Data Unit
  u8ModbusADU[u8ModbusADUSize++] = _u8MBSlave;
  u8ModbusADU[u8ModbusADUSize++] = u8MBFunction;

  switch(u8MBFunction)
  {
```

```
    case ku8MBReadInputRegisters:
      u8ModbusADU[u8ModbusADUSize++] = highByte(_u16ReadAddress);
      u8ModbusADU[u8ModbusADUSize++] = lowByte(_u16ReadAddress);
      u8ModbusADU[u8ModbusADUSize++] = highByte(_u16ReadQty);
      u8ModbusADU[u8ModbusADUSize++] = lowByte(_u16ReadQty);
      break;
  }

  // append CRC
  u16CRC = 0xFFFF;
  for (i = 0; i < u8ModbusADUSize; i++)
  {
    u16CRC = crc16_update(u16CRC, u8ModbusADU[i]);
  }
  u8ModbusADU[u8ModbusADUSize++] = lowByte(u16CRC);
  u8ModbusADU[u8ModbusADUSize++] = highByte(u16CRC);
  u8ModbusADU[u8ModbusADUSize] = 0;

  // flush receive buffer before transmitting request
  while (_serial->read() != -1);

  // transmit request
  if (_preTransmission)
  {
    _preTransmission();
  }
  for (i = 0; i < u8ModbusADUSize; i++)
  {
    _serial->write(u8ModbusADU[i]);
  }

  u8ModbusADUSize = 0;
  _serial->flush();    // flush transmit buffer
  if (_postTransmission)
  {
    _postTransmission();
  }

  // loop until we run out of time or bytes, or an error occurs
  u32StartTime = millis();
  while (u8BytesLeft && !u8MBStatus)
  {
    if (_serial->available())
    {
#if __MODBUSMASTER_DEBUG__
      digitalWrite(__MODBUSMASTER_DEBUG_PIN_A__, true);
#endif
      u8ModbusADU[u8ModbusADUSize++] = _serial->read();
      u8BytesLeft--;
#if __MODBUSMASTER_DEBUG__
      digitalWrite(__MODBUSMASTER_DEBUG_PIN_A__, false);
#endif
    }
    else
    {
#if __MODBUSMASTER_DEBUG__
      digitalWrite(__MODBUSMASTER_DEBUG_PIN_B__, true);
#endif
```

```
      if (_idle)
      {
        _idle();
      }
#if __MODBUSMASTER_DEBUG__
      digitalWrite(__MODBUSMASTER_DEBUG_PIN_B__, false);
#endif
    }

    // evaluate slave ID, function code once enough bytes have been read
    if (u8ModbusADUSize == 5)
    {
      // verify response is for correct Modbus slave
      if (u8ModbusADU[0] != _u8MBSlave)
      {
        u8MBStatus = ku8MBInvalidSlaveID;
        break;
      }

      // verify response is for correct Modbus function code (mask
exception bit 7)
      if ((u8ModbusADU[1] & 0x7F) != u8MBFunction)
      {
        u8MBStatus = ku8MBInvalidFunction;
        break;
      }

      // check whether Modbus exception occurred; return Modbus Exception
Code
      if (bitRead(u8ModbusADU[1], 7))
      {
        u8MBStatus = u8ModbusADU[2];
        break;
      }

      // evaluate returned Modbus function code
      switch(u8ModbusADU[1])
      {
        case ku8MBReadInputRegisters:
          u8BytesLeft = u8ModbusADU[2];
          break;
      }
    }
    if ((millis() - u32StartTime) > ku16MBResponseTimeout)
    {
      u8MBStatus = ku8MBResponseTimedOut;
    }
  }

  // verify response is large enough to inspect further
  if (!u8MBStatus && u8ModbusADUSize >= 5)
  {
    // calculate CRC
    u16CRC = 0xFFFF;
    for (i = 0; i < (u8ModbusADUSize - 2); i++)
    {
      u16CRC = crc16_update(u16CRC, u8ModbusADU[i]);
    }
```

```
    // verify CRC
    if (!u8MBStatus && (lowByte(u16CRC) != u8ModbusADU[u8ModbusADUSize -
2] ||
      highByte(u16CRC) != u8ModbusADU[u8ModbusADUSize - 1]))
    {
      u8MBStatus = ku8MBInvalidCRC;
    }
  }

  // disassemble ADU into words
  if (!u8MBStatus)
  {
    // evaluate returned Modbus function code
    switch(u8ModbusADU[1])
    {
      case ku8MBReadInputRegisters:
        // load bytes into word; response bytes are ordered H, L, H, L,
...
        for (i = 0; i < (u8ModbusADU[2] >> 1); i++)
        {
          if (i < ku8MaxBufferSize)
          {
            _u16ResponseBuffer[i] = word(u8ModbusADU[2 * i + 3],
u8ModbusADU[2 * i + 4]);
          }

          _u8ResponseBufferLength = i;
        }
        break;
    }
  }

  _u8TransmitBufferIndex = 0;
  u16TransmitBufferLength = 0;
  _u8ResponseBufferIndex = 0;
  return u8MBStatus;
}
```

Esta función es la que realiza el grueso de la comunicación.

Tal como se observa en el código concatena todo el mensaje a mandar por el puerto serial dentro de la variable u8ModbusADU[i]. Primero los datos pasados por parámetro anteriormente: dirección del esclavo, función, los dos bytes de la dirección de lectura y los dos bytes de la longitud de lectura.

Finalmente calcula el CRC (código de control de errores), lo concatena al resto del mensaje y lo manda byte a byte a través del puerto serie.

Una vez se ha mandado todo el mensaje, espera la respuesta y la evalúa. En caso de que haya algún error en el mensaje de respuesta lo retorna en la variable u8MBStatus.

Si la comunicación ha sido exitosa, guarda la respuesta en el array u16ResponseBuffer.

node.getResponseBuffer

```
uint16_t ModbusMaster::getResponseBuffer(uint8_t u8Index)
{
  if (u8Index < ku8MaxBufferSize)
  {
    return _u16ResponseBuffer[u8Index];
  }
  else
  {
    return 0xFFFF;
  }
}
```

Esta función permite acceder al buffer del mensaje de respuesta. El acceso se hace byte a byte. Indicando en el parámetro u8Index la posición del array a la que queremos acceder.

Para utilizar todas estas funciones y clases en nuestro programa lo primero que se hace es incluir la librería en el sketch de Arduino con la siguiente sentencia:

```
#include <ModbusMaster.h>
```

Ahora se definen los pines a través de los que se va a elegir si el del dispositivo actuará como master o slave. En nuestro proyecto se utilizan los pines digitales 2 y 3 de la placa Arduino.

```
#define MAX485_DE      3

#define MAX485_RE_NEG  2
```

Se instancia el objeto "node" a la clase ModbusMaster.

```
ModbusMaster node;
```

A continuación se definen dos funciones que configuran los dos posibles estados del dispositivo, en modo master y slave.

```
void preTransmission()

 { digitalWrite(MAX485_RE_NEG, 1);

  digitalWrite(MAX485_DE, 1);}

void postTransmission()

 { digitalWrite(MAX485_RE_NEG, 0);

  digitalWrite(MAX485_DE, 0);}
```

Una vez definido esto, se configuran las características de la comunicación dentro de la zona de setup() del programa . Se configuran los dos pins de control master/slave como outputs.

```
pinMode(MAX485_RE_NEG, OUTPUT);
```

```
pinMode(MAX485_DE, OUTPUT);
```

Se inicializa el dispositivo Arduino en modo slave y la comunicación a una velocidad de 9600 baudios.

```
digitalWrite(MAX485_RE_NEG, 0);

digitalWrite(MAX485_DE, 0);

Serial.begin(9600);
```

Se llama a la función begin de la clase ModbusMaster con ID = 1 y puerto serie = Serial.

```
node.begin(1, Serial);
```

Se llama a las funciones preTransmission y postTransmission para configurarlas con los parámetros previamente definidos.

```
node.preTransmission(preTransmission);

node.postTransmission(postTransmission);
```

Una vez configurada la comunicación se implementa dentro del ciclo de ejecución del programa.

```
//Funcion comunicacion con el analizador

void com_analizador() {

  uint8_t result;

  // Lee 20 (0x14h) registros empezando en 0x0000)

  result = node.readInputRegisters(0x00, 0x14);

  if (result == node.ku8MBSuccess)

  {  Serial.print(F("Comunicación con analizador OK -->"));

     registros--;

     v_dec = node.getResponseBuffer(0x01)/10.0f;

     i_dec = node.getResponseBuffer(0x03)/1000.0f;

     pactiva_dec = node.getResponseBuffer(0x05)*1.0f;

     fpot_dec = node.getResponseBuffer(0x09)/100000.0f;

     frec_dec = node.getResponseBuffer(0x0d)/10.0f;

     eact_dec = node.getResponseBuffer(0x13)/1.0f;

     suma_medias();}
```

```
  delay(1);

  }
```

En esta función lo primero que se hace es definir una variable “result” numérica de 1 byte sin signo, para evaluar el retorno de la función, y un array de 6 elementos con variables “data” numérica de 2 bytes sin signo.

Después se llama a la función readInputRegisters con la dirección del primer registro de lectura = 0x00 y cantidad de registros a leer = 0x14 (20 en decimal).

Los datos que nos envía el analizador de red (los de respuesta) se guardan dentro de la variable u16ResponseBuffer de la clase ModbusMaster. Para acceder a ellos, se ha de realizar una llamada a la función getResponseBuffer indicando el registro dentro del array de respuesta que queremos recuperar. Debemos tener en cuenta que el array se ha definido con variables de 16 bits (2 bytes), mientras que cada parámetro que nos retorna el analizador tiene una longitud de 32 bits (4 bytes).

Finalmente se guardan los datos recibidos del analizador en variables tipo int para tratarlas de manera más sencilla a lo largo del resto del proceso.

4.4. Ethernet

Como se ha descrito anteriormente, para dotar al proyecto de un entorno más gráfico se implementará una página web en lenguaje html para displayar los datos recibidos desde el analizador en tiempo real.

Para ello, se utiliza la shield Ethernet de Arduino y, a través de este protocolo de comunicación y utilizando alguno de los puertos disponibles del router de la escuela se enviarán los datos. Pudiéndose estos consultar desde cualquier ordenador o monitor conectado a la red.

Para el manejo de las funciones Ethernet se utiliza la librería Ethernet.h con la que cuenta la placa Arduino Mega. Simplemente tenemos que añadirla al sketch.

```
#include <Ethernet.h>
```

A continuación se detalla el proceso implementado para esto.

Primero se deben definir las direcciones MAC e IP que se usaran para la comunicación Ethernet y se selecciona el puerto del server para el HTTP.

```
// Introduzca una dirección MAC y la dirección IP para el controlador

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

IPAddress ip(192,168,1,50);

// Direccion IP de la subred de su Area Local y es la que

// usara para conectarse por el Navegador.

EthernetServer server = EthernetServer(80);      // Puerto 80 por defecto para HTTP
```

Inicializamos la conexión Ethernet dentro del área de setup:

```
//Inicializa la conexión Ethernet y el servidor

  Ethernet.begin(mac, ip);

  server.begin();
```

La comunicación Ethernet se inicia con las direcciones MAC e IP fijadas previamente.

Una vez configurada la comunicación se implementa dentro del ciclo de ejecución del programa.

```
 EthernetClient client = server.available(); //Cliente conectado al servidor

  if (client) {

    String request = "";

    while (client.available()) {

      char c = client.read();

      if (c == '\n') break;

      else request += c;

    }

    while (client.available()) {

      client.read();

    }

    if (request.startsWith("GET / HTTP")) envia_web(client);

    else     if     (request.startsWith("GET     /ajax     HTTP")) envia_ajax(client);

}
```

En caso de que haya solicitud de algún cliente y la petición sea de apertura de la página se abre el fichero que contiene la página web htmlweb.htm y se envía a través del puerto serial hasta final de fichero, utilizando la función write de la clase EthernetServer. Al finalizar de leer el fichero, este se cierra.

```
void envia_web(EthernetClient &client_web){

     // Abrimos la pÃ¡gina de la SD y enviamos la cabecera de la web

     File webfile = SD.open("htmlweb.htm");

      if (webfile) {

         while (webfile.available()) {

                client_web.write(webfile.read());

         }

        webfile.close();

             }
```

```
    // close the connection:

    client_web.flush();

    client_web.stop();

}
```

Posteriormente, en caso de que la petición sea de actualización, utilizando AJAX, se enviarán por el puerto solamente los datos a actualizar en la web, en este caso los datos de los parámetros recibidos del analizador de red.

```
void envia_ajax(EthernetClient &client_ajax){

  Serial.println(F("envia_ajax -->"));

  client_ajax.println(F("HTTP/1.1 200 OK"));

  client_ajax.println(F("Content-Type:            application/json;
charset=utf-8"));

  client_ajax.println();

  client_ajax.print(F("{\"voltaje\":\""));

  client_ajax.print(v_dec);

  client_ajax.print(F("\""));

  client_ajax.print(F(",\"intensidad\":\""));

  client_ajax.print(i_dec);

  client_ajax.print(F("\""));

  client_ajax.print(F(",\"potencia\":\""));

  client_ajax.print(pactiva_dec);

  client_ajax.print(F("\""));

  client_ajax.print(F(",\"frecuencia\":\""));

  client_ajax.print(frec_dec);

  client_ajax.print(F("\""));

  client_ajax.print(F(",\"fpot\":\""));

  client_ajax.print(fpot_dec);

  client_ajax.print(F("\""));
```

```
  client_ajax.print(F(",\"eact\":\""));

  client_ajax.print(eact_dec);

  client_ajax.println(F("\"}"));

    // close the connection:

  delay(1);

  client_ajax.flush();

  client_ajax.stop();

}
```

Vemos que el mensaje json en el que se mandan los datos es el siguiente:

```
{"voltaje":"v_dec”,"intensidad":"i_dec”,"potencia":"pactiva_dec”
,"frecuencia":"frec_dec” ,"fpot":"fpot_dec” ,"eact":"eact_dec”}
```

4.5. Registro en tarjeta SD

La placa shield Ethernet de arduino cuenta con una ranura para tarjeta micro SD. Ésta será utilizada para registrar los datos provenientes del analizador de red.

Para el manejo de la tarjeta se utiliza la librera SD.h, integrándola en el sketch.

Dado que la comunicación con el analizador se realiza aproximadamente cada segundo (pensando sobre todo en la visualización en tiempo real a través de la web), se considera innecesario registrar los datos en la tarjeta SD con la misma frecuencia. Por lo tanto se generará una media en periodos de 3 minutos que será la que se grabará en el registro.

De esta manera ahorraremos espacio de memoria en la tarjeta, además de rebajar sustancialmente la carga de trabajo de la placa, ya que únicamente llamará a las funciones de la librería SD.h una vez cada 3 minutos.

Se ha diseñado el aspecto de los registros tal como se muestra en la **Figura 4.13**:

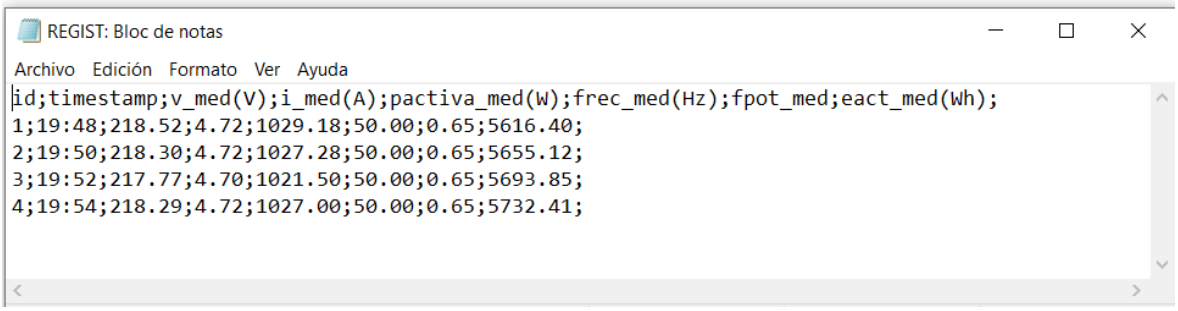


Figura 4.13. Registro tarjeta SD (Fuente: Juan Carlos Ruiz).

Vemos que el fichero cuenta con un registro de cabecera y N registros de datos con sus respectivos campos separados por punto y coma. De esta manera los registros serán más fáciles de exportar a formato .csv (Excel) cuando se desee.

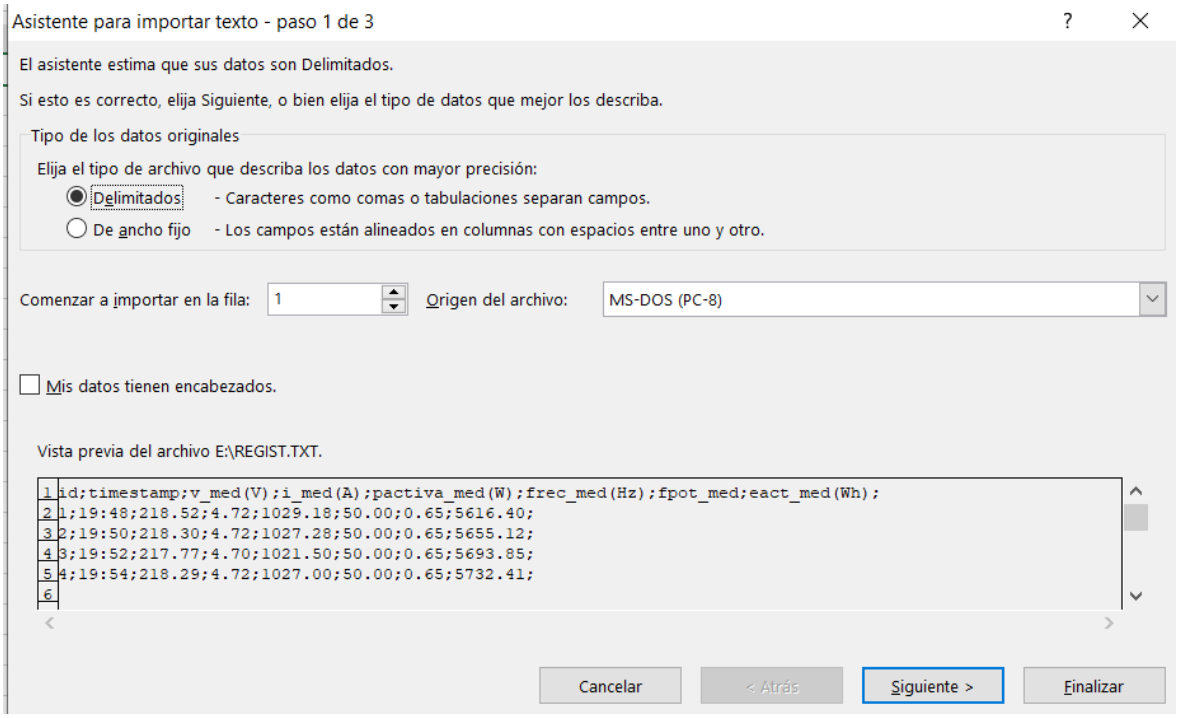


Figura 4.14. Exportar datos .csv (Fuente: Juan Carlos Ruiz).

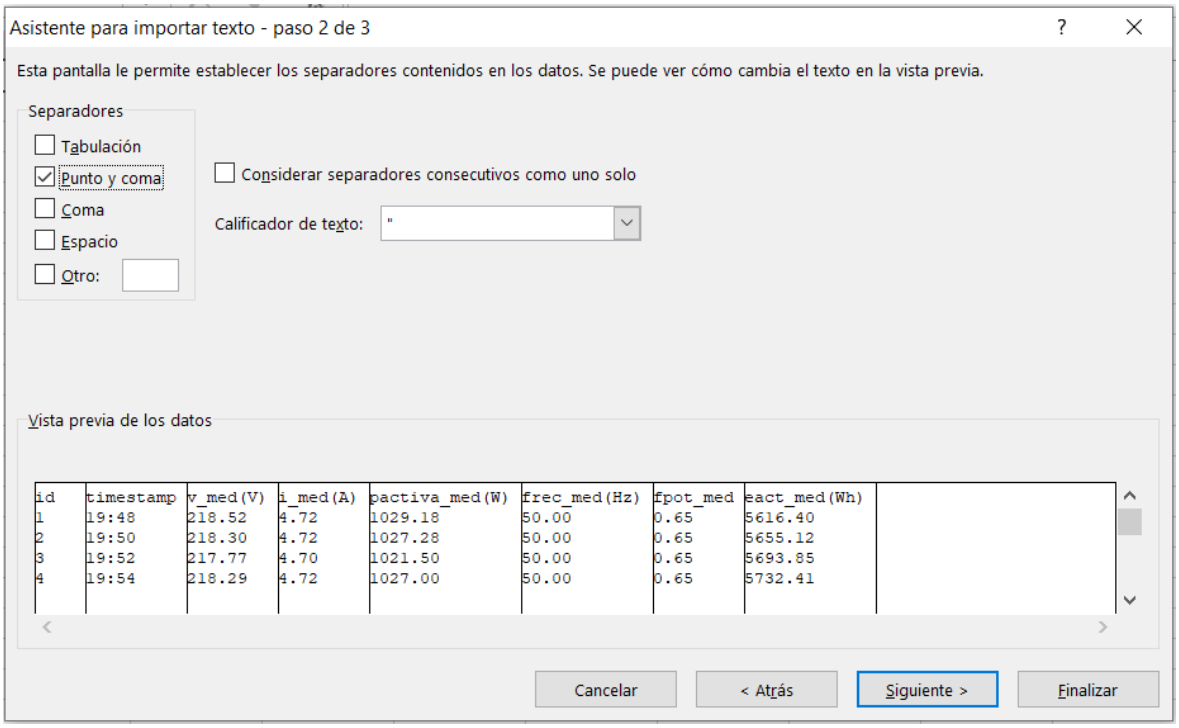


Figura 4.15. Columnas registros .csv (Fuente: Juan Carlos Ruiz).

	A	B	C	D	E	F	G	H	I
1	id	timestamp	v_med(V)	i_med(A)	pactiva_med(W)	frec_med(Hz)	fpot_med	eact_med(Wh)	
2	1	19:48	218.52	4.72	1029.18	50.00	0.65	5616.40	
3	2	19:50	218.30	4.72	1027.28	50.00	0.65	5655.12	
4	3	19:52	217.77	4.70	1021.50	50.00	0.65	5693.85	
5	4	19:54	218.29	4.72	1027.00	50.00	0.65	5732.41	
6									
7									
8									

Figura 4.16. Registro de datos exportados a excel (Fuente: Juan Carlos Ruiz).

En la **Figura 4.16** vemos el aspecto final de los registros exportados. A partir de este, se podrían analizar los datos a través de gráficos o demás herramientas visuales.

4.6. WEB

Como se ha comentado anteriormente se va a dotar al proyecto de un entorno un poco más gráfico para mostrar los datos recibidos del analizador en tiempo real a través de la red de la escuela.

Con la intención de que puedan ser consultados y mostrados desde cualquier dispositivo conectado a la red, ahorrando así el trabajo de tener que implementar conexiones VGA.

De esta manera, solo bastará un monitor (o varios) conectados a la red, de la misma manera que los que hay actualmente instalados en la EEBE, y un navegador web para poder visualizar los datos.

La web se mandará a través de la comunicación Ethernet implementada, utilizando las funciones de la librería Ethernet.h.

Por comodidad y para que el código .ino cargado en la placa sea lo más limpio posible se ha optado por guardar el grueso del código html en un fichero "htmlweb.htm" dentro de la tarjeta SD.

Así, se leerá el fichero y se mandará vía Ethernet hasta su último carácter.

Una vez cargada la página por primera vez, se mandarán los datos de actualizacion usando AJAX desde el código .ino, usando la función print. Los datos de la página se refrescarán cada 5 segundos.

A continuación se muestran las partes más relevantes del código html:

```
<style "type=text/css">
      * {margin: 0; padding: 0; background-color: #2D89FF;}
      div {display: inline-block; background-color: blue; color: white;
border: 1px solid black; padding: 5px; margin: 5px; height: auto;
      width: 450px; border-radius: 10px; text-align: center; float: left;
font-weight:bold;}
      #principal {background-color: #2D89FF;}
      h1 { padding: 0; display: inline-block; background-color: #2D89FF;
color: white; width: 100%;}
      h2 { padding: 0; display: inline-block; background-color: #2D89FF;
color: white; width: 100%;}
      h3 { padding: 0; display: inline-block; background-color: #2D89FF;
width: 100%;}
      p  { padding: 0; display: inline-block; border: #2D89FF 2px solid;
background-color: #2D89FF; color: white; width: 25%;}

</style>
```

Se define el estilo de la página, en esta predominaran los colores corporativos de la Universitat Politècnica de Catalunya, usados también en la página web de la escuela, el blanco y el azul.

En el primer script se define el estilo de los indicadores de datos que se van a utilizar, en este caso unos gauges con un display digital además del indicador de aguja, el aspecto se muestra en la **Figura 4.17**.

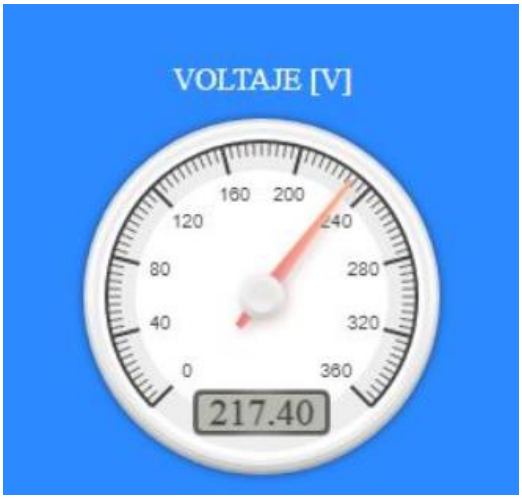


Figura 4.17. Indicador de voltaje (Fuente: Juan Carlos Ruiz).

```
<script type="text/javascript">
 var voltaje;
 var intensidad;
 var potencia;
 var frecuencia;
 var fpot;
 var eact;

function update() {  <!--esta funcion es la que me interesa --> recoge
los datos del arduino -->
 var xhttp;
  xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
        var json = JSON.parse(xhttp.responseText);
        voltaje = json.voltaje;
        intensidad = json.intensidad;
        potencia = json.potencia;
        fpot = json.fpot;
        frecuencia = json.frecuencia;
        eact = json.eact;

    }
      };
  xhttp.open("GET", "http://192.168.1.50/ajax", true); <!-- la ip por
donde se mandan los datos y la peticion de datos -->
  xhttp.send();
   }
 var id_interval = setInterval('update()',5000);
</script>
```

En este segundo script se definen las funciones Json del documento. Se recogen los datos de la respuesta y se guardan en las variables globales de los diferentes parámetros: voltaje, intensidad, potencia, fpot, frecuencia y eact. Para posteriormente actualizarlos en el gauge. Seleccionamos un intervalo de petición de datos de 5 segundos.

```
<p> VOLTAJE [V]</p> <p> INTENSIDAD [A]</p> <p> POTENCIA [W]</p><br />

<p><canvas id="an_gauge_1" data-major-ticks="0 40 80 120 160 200 240 280 320 360" data-type="canv-gauge" data-min-value="0" data-max-value="360" data-onready="setInterval( function() { Gauge.Collection.get('an_gauge_1').setValue(voltaje);}, 200);"></canvas></p>

<p><canvas id="an_gauge_2" data-major-ticks="0 2 4 6 8 10 12 14 16 18 20" data-type="canv-gauge" data-min-value="0" data-max-value="20" data-onready="setInterval( function() { Gauge.Collection.get('an_gauge_2').setValue(intensidad);}, 200);" ></canvas></p>

<p><canvas id="an_gauge_3" data-major-ticks="0 500 1000 1500 2000 2500 3000 3500 4000 4500 5000" data-type="canv-gauge" data-min-value="0" data-max-value="5000" data-onready="setInterval( function() { Gauge.Collection.get('an_gauge_3').setValue(potencia);}, 200);" ></canvas></p>

<p> FRECUENCIA [Hz]</p> <p> FACTOR DE POTENCIA </p> <p> ENERGIA ACTIVA [W.h]</p><br />

<p><canvas id="an_gauge_4" data-major-ticks="0 10 20 30 40 50 60 70 80 90 100" data-type="canv-gauge" data-min-value="0" data-max-value="100" data-onready="setInterval( function() { Gauge.Collection.get('an_gauge_4').setValue(frecuencia);}, 200);" ></canvas></p>

<p><canvas id="an_gauge_5" data-major-ticks="0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1" data-type="canv-gauge" data-min-value="0" data-max-value="1" data-onready="setInterval( function() { Gauge.Collection.get('an_gauge_5').setValue(fpot);}, 200);" ></canvas></p>

<p><canvas id="an_gauge_6" data-major-ticks="0 1000 2000 3000 4000 5000 6000 7000 8000 9000 10000" data-type="canv-gauge" data-min-value="0" data-max-value="10000" data-onready="setInterval( function() { Gauge.Collection.get('an_gauge_6').setValue(eact);}, 200);" ></canvas></p>
```

Aquí es donde definimos las características propias de cada gauge.

Se define para cada indicador el rango de valores a mostrar y se selecciona como valor a mostrar el dato recibido anteriormente.

El aspecto final de la web es el mostrado en la **Figura 4.18**:

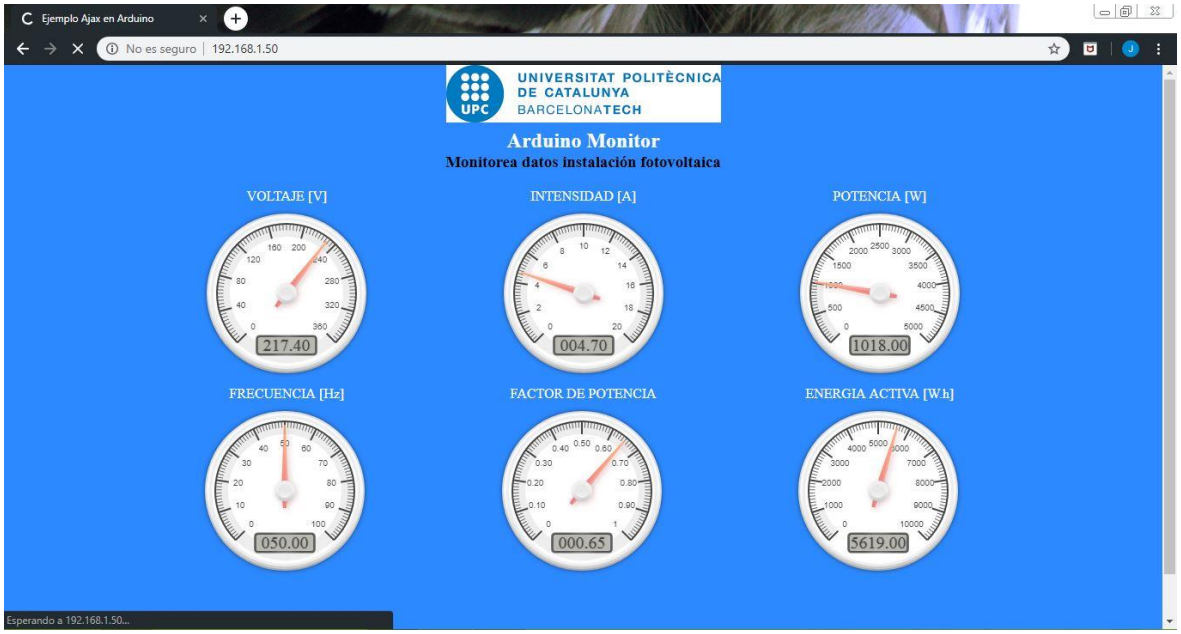


Figura 4.18. Página Web monitor de datos (Fuente: Juan Carlos Ruiz).

4.7. Hora Udp

Con la intención de poder analizar los datos recogidos en la tarjeta SD de forma más ordenada y poder establecer una relación con las distintas franjas horarias del día, se obtendrá la hora actual desde la red.

Para el manejo de las funciones para recoger la hora de la red se utiliza la librería EthernetUdp.h con la que cuenta la placa Arduino Mega. Simplemente tenemos que añadirla al sketch.

```
#include <EthernetUdp.h>
```

A continuación se detalla el proceso implementado para esto.

Primero se deben definir la dirección IP que se usaran para la comunicación y se crea la instancia del server para el Udp.

```
IPAddress timeServer(193,92,150,3);     // time.nist.gov NTP server
const int NTP_PACKET_SIZE = 48;         // La hora son los primeros 48
bytes del mensaje
byte packetBuffer[ NTP_PACKET_SIZE];      // buffer para los paquetes
EthernetUDP Udp;                 // Una instancia de UDP para enviar y
recibir mensajes
```

Inicializamos la conexión Ethernet dentro del área de setup:

```
Udp.begin(80);
```

Una vez configurada la comunicación se implementa dentro del ciclo de ejecución del programa.

```
void com_udp_ntp() {
  sendNTPpacket(timeServer);   // Enviar una petición al servidor NTP
  delay(100);                  // Damos tiempo a la respuesta
  if ( Udp.parsePacket() ) {
    Udp.read(packetBuffer,NTP_PACKET_SIZE);  // Leemos el paquete

// La hora empieza en el byte 40 de la respuesta y es de 4 bytes o 2
words
// Hay que extraer ambas words:
    unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
    unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
    // se combinan los 4 bytes (2 words) en una variable long integer
    // this is NTP time (seconds since Jan 1 1900):
    unsigned long secsSince1900 = highWord << 16 | lowWord;
    // Ahora a convertir el tiempo NTP en tiempo Uxix:
    // Unix time empieza en Jan 1 1970. En segundos, son 2208988800:
    const unsigned long seventyYears = 2208988800UL;
    // extrae setenta years:
    unsigned long epoch = secsSince1900 - seventyYears;
    horas = ((epoch  % 86400L) / 3600) + 2;   // Horas (86400 segundos
por dia)
    minutos = ((epoch  % 3600) / 60); // Minutos:
  }
  delay(100);
}
```

```
unsigned long sendNTPpacket(IPAddress& address)
{
  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;

  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:

  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer,NTP_PACKET_SIZE);
  Udp.endPacket();
//}
```

Basicamente lo que hacemos en este punto es acceder a una URL que contiene el timestamp actual y le mandamos una request para que nos conteste con los datos solicitados. Una vez recibida la respuesta y guardada en el buffer, recuperamos la hora exacta del acceso.

4.8. Montaje de la placa e integración del circuito

Para integrar nuestro proyecto del circuito eléctrico de la EEBE se ha decidido integrar todos los elementos del montaje dentro de una caja mecanizada de la marca RETEX serie 102.

Figura 4.19. Caja RETEX serie 102 (Fuente: https://diotronic.com).

Características:

Las dimensiones de la caja son 155 x 95 x 60 mm.

Caja fabricada en ABS con tapa en policarbonato transparente.

Rebaje en tapa y base para teclados de membrana.

Torretas en base para fijación de circuitos.

Base resistente hasta 85ºC. Tapa hasta 120ºC.

Además se alimentará la placa a través de un transformador de 230V a 9 V de 1A.

Figura 4.20. Transformador a 9V (Fuente: https://diotronic.com).

La caja se ha mecanizado para que la placa Arduino y sus shields encajen correctamente y queden fijadas, además de permitir las conexiones.

El aspecto del montaje final se muestra a continuación:

Figura 4.21. Montaje Final 1 (Fuente: Juan Carlos Ruiz).

Figura 4.22. Montaje Final 2 (Fuente: Juan Carlos Ruiz).

La conexión entre la shield RS485 y el analizador de red se hará a través de un par trenzado de dos hilos, a ser posible con apantallamiento para evitar posibles interferencias. Sobre todo si la distancia entre los dos dispositivos es larga o si la red va a pasar por zonas con interferencias electromagnéticas.

Para la conexión Ethernet se usará un cable Ethernet estándar del tipo CAT5 o CAT6 con conectores RJ45.

La caja mecanizada cuenta con agujeros para, una vez decidida la ubicación del dispositivo, fijarla a la pared si se requiere. La longitud de los cables de conexión (Ethernet, par trenzado) dependerá del emplazamiento de los componentes.

En cuanto al cable de alimentación de la placa Arduino tiene una longitud aproximada de 1,5 metros. Esto tampoco es un gran problema a la hora de elegir la ubicación del sistema dado que no haber ningún enchufe cerca, bastaría con un simple alargo.

5. Software

5.1. Sketch Arduino

```
// Librerias usadas
#include <Ethernet.h>
#include <SD.h>
#include <ModbusMaster.h>
#include <EthernetUdp.h>

// Declaracion de ficheros
File dataFile;
File webfile;

// Parametros y variables comunicacion RS485
#define MAX485_DE      3
#define MAX485_RE_NEG  2

void preTransmission()
{ digitalWrite(MAX485_RE_NEG, 1);
  digitalWrite(MAX485_DE, 1);
}

void postTransmission()
{ digitalWrite(MAX485_RE_NEG, 0);
  digitalWrite(MAX485_DE, 0);
}


// Inicializa objeto ModbusMaster
ModbusMaster node;

// Parametros y variables comunicacion Ethernet
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Direccion MAC por defecto
IPAddress ip(192, 168, 1, 50); // Direccion IP de la subred del Area Local que usara para
conectarse por el Navegador.
EthernetServer server = EthernetServer(80);    // Puerto 80 por defecto para HTTP

// Variables decimales de los datos del analizador
float v_dec, i_dec, pactiva_dec, frec_dec, fpot_dec, eact_dec;

// Variables decimales media de los datos del analizador
float v_med, i_med, pactiva_med, frec_med, fpot_med, eact_med;

// Variables ethernet udp hora y fecha
IPAddress timeServer(193, 92, 150, 3);  // time.nist.gov NTP server (fallback)
const int NTP_PACKET_SIZE = 48;       // La hora son los primeros 48 bytes del mensaje
byte packetBuffer[ NTP_PACKET_SIZE];   // buffer para los paquetes de la comunicacion
EthernetUDP Udp;                // Una instancia de UDP para enviar y recibir mensajes

// Otras variables
byte registros = 150;
byte id = 0;
int horas, minutos ;
String puntocoma = ";" ;
String dospuntos = ":" ;


////////////////////////////////////////////////////////////////////////////////

//Configuracion inicial
void setup() {
// Comprueba que la targeta esta conectada y graba registro de cabecera
  if (!SD.begin(4))
        { Serial.println(F("No hay tarjeta"));
         return;
```

```
        }
  pinMode(4, OUTPUT);
  graba_cabecera_sd();

// Inicializa la conexion Ethernet y el servidor
  pinMode(10, OUTPUT);
  Ethernet.begin(mac, ip);
  server.begin();

// Configuracion inicial Modbus a 9600 bauds , esclavo ID 1
  pinMode(MAX485_RE_NEG, OUTPUT);
  pinMode(MAX485_DE, OUTPUT);
  digitalWrite(MAX485_RE_NEG, 0);
  digitalWrite(MAX485_DE, 0);
  Serial.begin(9600);
  node.begin(1, Serial);

// Callbacks para configurar la comunicacion RS485 correctamente
  node.preTransmission(preTransmission);
  node.postTransmission(postTransmission);

// Inicializa el servidor udp
  Udp.begin(80);
}


// Proceso
void loop() {
  com_udp_ntp();
  delay(200);
  com_analizador();
  if (registros == 0) {
    calcula_media();
    graba_sd();
    inicializa_medias();
     }
// Envia datos a la red ip configurada en el inicio
  EthernetClient client = server.available(); //Cliente conectado al servidor
  if (client) {
    Serial.print(F("Cliente conectado -->"));
    String request = "";
    while (client.available()) {
      char c = client.read();
      if (c == '\n') break;
      else request += c;
       }

    while (client.available()) {
      client.read();
          }
    Serial.print(F("Request:"));
    Serial.print(request);
    if (request.startsWith("GET / HTTP")) envia_web(client);
    else if (request.startsWith("GET /ajax HTTP")) envia_ajax(client);
  }

  Serial.print(F("Fin programa :"));
  Serial.println(registros);
  delay(100);
}

// Comunicacion udp para recuperar fecha y hora
void com_udp_ntp() {
  Serial.print(F("Recupera hora: "));
  sendNTPpacket(timeServer);   // Enviar una peticion al servidor NTP
  delay(100);                  // Damos tiempo a la respuesta
  if ( Udp.parsePacket() ) {
    Udp.read(packetBuffer, NTP_PACKET_SIZE); // Leemos el paquete

// La hora empieza en el byte 40 de la respuesta y es de 4 bytes o 2 words
    unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
    unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
```

```
// Este es el NTP time (segundos desde Jan 1 1900):
    unsigned long secsSince1900 = highWord << 16 | lowWord;
// Se convierte el tiempo NTP en tiempo Uxix:
// Unix time empieza en Jan 1 1970. En segundos, son 2208988800:
    const unsigned long seventyYears = 2208988800UL;
// Resta 70 años:
    unsigned long epoch = secsSince1900 - seventyYears;
    horas = ((epoch  % 86400L) / 3600) + 1;   // Horas (86400 segundos por dia)
    minutos = ((epoch  % 3600) / 60); // Minutos:
  }
  delay(10);
  Serial.print(horas);
  Serial.print(dospuntos);
  Serial.print(minutos);
  Serial.print(F(" -->"));
}

unsigned long sendNTPpacket(IPAddress& address)
{
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;

  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:

  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer, NTP_PACKET_SIZE);
  Udp.endPacket();
}

// Funcion comunicacion con el analizador
void com_analizador() {
  uint8_t result;
// Lee 20 (0x14h) registros empezando en 0x0000)
  result = node.readInputRegisters(0x00, 0x14);
  if (result == node.ku8MBSuccess)
  { Serial.print(F("Comunicacion con analizador OK -->"));
    registros--;
    v_dec = node.getResponseBuffer(0x01) / 10.0f;
    i_dec = node.getResponseBuffer(0x03) / 1000.0f;
    pactiva_dec = node.getResponseBuffer(0x05) * 1.0f;
    fpot_dec = node.getResponseBuffer(0x09) / 100000.0f;
    frec_dec = node.getResponseBuffer(0x0d) / 10.0f;
    eact_dec = node.getResponseBuffer(0x13) / 1.0f;
    suma_medias();
  }
  delay(1);
}

// Funcion suma medias
void suma_medias() {
  v_med += v_dec;
  i_med += i_dec;
  pactiva_med += pactiva_dec;
  frec_med += frec_dec;
  fpot_med += fpot_dec;
  eact_med += eact_dec;
}

// Funcion calcula media
void calcula_media()  {
  v_med = v_med / 150;
  i_med = i_med / 150;
```

```
  pactiva_med = pactiva_med / 150;
  frec_med = frec_med / 150;
  fpot_med = fpot_med / 150;
  eact_med = eact_med / 150;
}

// Inicializa medias
void inicializa_medias()  {
  v_med = 0;
  i_med = 0;
  pactiva_med = 0;
  frec_med = 0;
  fpot_med = 0;
  eact_med = 0;
}

// Funciones de grabacion en la targeta SD
void graba_cabecera_sd() {
// Comprueba si existe el fichero
  dataFile = SD.open("regist.txt", FILE_WRITE); // Define abrir el fichero en modo
grabacion
  if (dataFile)   // Si ha podido abrir el fichero graba el registro de cabecera
  { if (SD.exists("regist.txt"))
    {

dataFile.println(F("id;timestamp;v_med(V);i_med(mA);pactiva_med(mW);frec_med(Hz);fpot_med(x
100);eact_med(Wh);"));
      dataFile.flush();
      dataFile.close();
      Serial.print(F("Graba registro de cabecera -->"));
    }
  }
  delay(1);
}

void graba_sd() {
  //Comprueba si existe el fichero
  dataFile = SD.open("regist.txt", FILE_WRITE);
  if (dataFile)   // Si ha podido abrir el fichero graba el registro de datos
  { if (SD.exists("regist.txt"))
    {
      id ++;
      String registrosd = id + puntocoma + horas + dospuntos + minutos + puntocoma + v_med
+ puntocoma + i_med + puntocoma + pactiva_med + puntocoma + frec_med + puntocoma + fpot_med
+ puntocoma + eact_med + puntocoma ;
      dataFile.println(registrosd);
      dataFile.flush();
      dataFile.close();
      Serial.print(F("Graba registro -->"));
    }
  }
  registros = 150;
  delay(1);
}

// Funciones de envio de la pagina web
void envia_web(EthernetClient &client_web) {
  // Abrimos la pagina de la SD y enviamos la cabecera de la web
  Serial.print(F("envia_web -->"));
  client_web.println(F("HTTP/1.1 200 OK"));
  client_web.println(F("Content-Type: text/html"));
  client_web.println(F("Connection: keep-alive"));
  client_web.println();
  webfile = SD.open("htmlweb.htm");
  if (webfile) {
    while (webfile.available()) {
      client_web.write(webfile.read());
    }
    delay(1);
    webfile.close();
  }
  // close the connection:
```

```
  delay(1);
  client_web.flush();
  client_web.stop();
}

void envia_ajax(EthernetClient &client_ajax) {
  Serial.println(F("envia_ajax -->"));
  client_ajax.println(F("HTTP/1.1 200 OK"));
  client_ajax.println(F("Content-Type: application/json; charset=utf-8"));
  client_ajax.println();
  client_ajax.print(F("{\"voltaje\":\""));
  client_ajax.print(v_dec);
  client_ajax.print(F("\""));
  client_ajax.print(F(",\"intensidad\":\""));
  client_ajax.print(i_dec);
  client_ajax.print(F("\""));
  client_ajax.print(F(",\"potencia\":\""));
  client_ajax.print(pactiva_dec);
  client_ajax.print(F("\""));
  client_ajax.print(F(",\"frecuencia\":\""));
  client_ajax.print(frec_dec);
  client_ajax.print(F("\""));
  client_ajax.print(F(",\"fpot\":\""));
  client_ajax.print(fpot_dec);
  client_ajax.print(F("\""));
  client_ajax.print(F(",\"eact\":\""));
  client_ajax.print(eact_dec);
  client_ajax.println(F("\"}"));
  // close the connection:
  delay(1);
  client_ajax.flush();
  client_ajax.stop();
}
```

5.2. Html

```
<!DOCTYPE html>

<html>

<head>

<title>Ejemplo Ajax en Arduino</title>

<style "type=text/css">

	* {margin: 0; padding: 0; background-color: #2D89FF;}

	div {display: inline-block; background-color: blue; color: white; border: 1px solid black; padding: 5px; margin: 5px; height: auto;

	width: 450px; border-radius: 10px; text-align: center; float: left; font-weight:bold;}

	#principal {background-color: #2D89FF;}

	h1 { padding: 0; display: inline-block; background-color: #2D89FF; color: white; width: 100%;}

	h2 { padding: 0; display: inline-block; background-color: #2D89FF; color: white; width: 100%;}

	h3 { padding: 0; display: inline-block; background-color: #2D89FF; width: 100%;}

	p  { padding: 0; display: inline-block; border: #2D89FF 2px solid; background-color: #2D89FF; color: white; width: 25%;}



</style>

<script>

<!-- Gauge Code Starts -->

var Gauge=function(b){function l(a,b){for(var c in b)"object"==typeof b[c]&&"[object Array]"!==Object.prototype.toString.call(b[c])&&"renderTo"!=c?("object"!=typeof a[c]&&(a[c]={}),l(a[c],b[c])):a[c]=b[c]}function q(){z.width=b.width;z.height=b.height;A=z.cloneNode(!0);B=A.getContext("2d");C=z.width;D=z.height;t=C/2;u=D/2;f=t<u?t:u;A.i8d=!1;B.translate(t,u);B.save();a.translate(t,u);a.save()}function v(a){var b=new Date;G=setInterval(function(){var c=(new Date-b)/a.duration;1<c&&(c=1);var f=("function"==

typeof a.delta?a.delta:M[a.delta])(c);a.step(f);1==c&&clearInterval(G)},a.delay||10)}function k(){G&&clearInterval(G);var a=I-n,h=n,c=b.animation;v({delay:c.delay,duration:c.duration,delta:c.fn,step:function(b){n=parseFloat(h)+a*b;E.draw()}})}function e(a){return a*Math.PI/180}function
```

```
g(b,h,c){c=a.createLinearGradient(0,0,0,c);c.addColorStop(0,b);c.addColorStop(1,h);return
c}function p(){var m=93*(f/100),h=f-
m,c=91*(f/100),e=88*(f/100),d=85*(f/100);a.save();b.glow&&(a.shadowBlur=h,a.shadowColor=

"rgba(0, 0, 0,
0.5)");a.beginPath();a.arc(0,0,m,0,2*Math.PI,!0);a.fillStyle=g("#ddd","#aaa",m);a.fill();a.
restore();a.beginPath();a.arc(0,0,c,0,2*Math.PI,!0);a.fillStyle=g("#fafafa","#ccc",c);a.fil
l();a.beginPath();a.arc(0,0,e,0,2*Math.PI,!0);a.fillStyle=g("#eee","#f0f0f0",e);a.fill();a.
beginPath();a.arc(0,0,d,0,2*Math.PI,!0);a.fillStyle=b.colors.plate;a.fill();a.save()}functi
on w(a){var
h=!1;a=0===b.majorTicksFormat.dec?Math.round(a).toString():a.toFixed(b.majorTicksFormat.dec
);return 1<b.majorTicksFormat["int"]?

(h=-1<a.indexOf("."),-1<a.indexOf("-")?"-
"+(b.majorTicksFormat["int"]+b.majorTicksFormat.dec+2+(h?1:0)-a.length)+a.replace("-
","")":""+(b.majorTicksFormat["int"]+b.majorTicksFormat.dec+1+(h?1:0)-
a.length)+a):a}function d(){var
m=81*(f/100);a.lineWidth=2;a.strokeStyle=b.colors.majorTicks;a.save();if(0===b.majorTicks.l
ength){for(var h=(b.maxValue-
b.minValue)/5,c=0;5>c;c++)b.majorTicks.push(w(b.minValue+h*c));b.majorTicks.push(w(b.maxVal
ue))}for(c=0;c<b.majorTicks.length;++c)a.rotate(e(45+c*(270/(b.majorTicks.length-

1)))),a.beginPath(),a.moveTo(0,m),a.lineTo(0,m-
15*(f/100)),a.stroke(),a.restore(),a.save();b.strokeTicks&&(a.rotate(e(90)),a.beginPath(),a
.arc(0,0,m,e(45),e(315),!1),a.stroke(),a.restore(),a.save())}function J(){var
m=81*(f/100);a.lineWidth=1;a.strokeStyle=b.colors.minorTicks;a.save();for(var
h=b.minorTicks*(b.majorTicks.length-
1),c=0;c<h;++c)a.rotate(e(45+c*(270/h))),a.beginPath(),a.moveTo(0,m),a.lineTo(0,m-
7.5*(f/100)),a.stroke(),a.restore(),a.save()}function s(){for(var
m=55*(f/100),h=0;h<b.majorTicks.length;++h){var c=

F(m,e(45+h*(270/(b.majorTicks.length-1))));a.font=20*(f/200)+"px
Arial";a.fillStyle=b.colors.numbers;a.lineWidth=0;a.textAlign="center";a.fillText(b.majorTi
cks[h],c.x,c.y+3)}}function x(a){var
h=b.valueFormat.dec,c=b.valueFormat["int"];a=parseFloat(a);var
f=0>a;a=Math.abs(a);if(0<h){a=a.toFixed(h).toString().split(".");h=0;for(c-
=a[0].length;h<c;++h)a[0]="0"+a[0];a=(f?"-
":"")+a[0]+"."+a[1]}else{a=Math.round(a).toString();h=0;for(c-
=a.length;h<c;++h)a="0"+a;a=(f?"-":"")+a}return a}function F(a,b){var c=

Math.sin(b),f=Math.cos(b);return{x:0*f-a*c,y:0*c+a*f}}function N(){a.save();for(var
m=81*(f/100),h=m-15*(f/100),c=0,g=b.highlights.length;c<g;c++){var
d=b.highlights[c],r=(b.maxValue-b.minValue)/270,k=e(45+(d.from-b.minValue)/r),r=e(45+(d.to-
b.minValue)/r);a.beginPath();a.rotate(e(90));a.arc(0,0,m,k,r,!1);a.restore();a.save();var
l=F(h,k),p=F(m,k);a.moveTo(l.x,l.y);a.lineTo(p.x,p.y);var
p=F(m,r),n=F(h,r);a.lineTo(p.x,p.y);a.lineTo(n.x,n.y);a.lineTo(l.x,l.y);a.closePath();a.fil
lStyle=d.color;a.fill();

a.beginPath();a.rotate(e(90));a.arc(0,0,h,k-
0.2,r+0.2,!1);a.restore();a.closePath();a.fillStyle=b.colors.plate;a.fill();a.save()}}funct
ion K(){var
m=12*(f/100),h=8*(f/100),c=77*(f/100),d=20*(f/100),k=4*(f/100),r=2*(f/100),l=function(){a.s
hadowOffsetX=2;a.shadowOffsetY=2;a.shadowBlur=10;a.shadowColor="rgba(188, 143, 143,
0.45)"};l();a.save();n=0>n?Math.abs(b.minValue-n):0<b.minValue?n-
```

```
b.minValue:Math.abs(b.minValue)+n;a.rotate(e(45+n/((b.maxValue-
b.minValue)/270)));a.beginPath();a.moveTo(-r,-d);a.lineTo(-k,

0);a.lineTo(-1,c);a.lineTo(1,c);a.lineTo(k,0);a.lineTo(r,-
d);a.closePath();a.fillStyle=g(b.colors.needle.start,b.colors.needle.end,c-
d);a.fill();a.beginPath();a.lineTo(-0.5,c);a.lineTo(-1,c);a.lineTo(-k,0);a.lineTo(-r,-
d);a.lineTo(r/2-2,-d);a.closePath();a.fillStyle="rgba(255,            255,            255,
0.2)";a.fill();a.restore();l();a.beginPath();a.arc(0,0,m,0,2*Math.PI,!0);a.fillStyle=g("#f0
f0f0","#ccc",m);a.fill();a.restore();a.beginPath();a.arc(0,0,h,0,2*Math.PI,!0);a.fillStyle=
g("#e8e8e8","#f5f5f5",h);a.fill()}

function     L(){a.save();a.font=40*(f/200)+"px     Led";var     b=x(y),h=a.measureText("-
"+x(0)).width,c=f-33*(f/100),g=0.12*f;a.save();var                  d=-h/2-0.025*f,e=c-g-
0.04*f,h=h+0.05*f,g=g+0.07*f,k=0.025*f;a.beginPath();a.moveTo(d+k,e);a.lineTo(d+h-
k,e);a.quadraticCurveTo(d+h,e,d+h,e+k);a.lineTo(d+h,e+g-k);a.quadraticCurveTo(d+h,e+g,d+h-
k,e+g);a.lineTo(d+k,e+g);a.quadraticCurveTo(d,e+g,d,e+g-
k);a.lineTo(d,e+k);a.quadraticCurveTo(d,e,d+k,e);a.closePath();d=a.createRadialGradient(0,c
-0.12*f-0.025*f+(0.12*f+0.045*f)/

2,f/10,0,c-0.12*f-
0.025*f+(0.12*f+0.045*f)/2,f/5);d.addColorStop(0,"#888");d.addColorStop(1,"#666");a.strokeS
tyle=d;a.lineWidth=0.05*f;a.stroke();a.shadowBlur=0.012*f;a.shadowColor="rgba(0,    0,    0,
1)";a.fillStyle="#babab2";a.fill();a.restore();a.shadowOffsetX=0.004*f;a.shadowOffsetY=0.00
4*f;a.shadowBlur=0.012*f;a.shadowColor="rgba(0,                    0,                    0,
0.3)";a.fillStyle="#444";a.textAlign="center";a.fillText(b,-
0,c);a.restore()}Gauge.Collection.push(this);this.config={renderTo:null,width:200,height:20
0,title:!1,

maxValue:100,minValue:0,majorTicks:[],minorTicks:10,strokeTicks:!0,units:!1,valueFormat:{"i
nt":3,dec:2},majorTicksFormat:{"int":1,dec:0},glow:!0,animation:{delay:10,duration:250,fn:"
cycle"},colors:{plate:"#fff",majorTicks:"#444",minorTicks:"#666",title:"#888",units:"#888",
numbers:"#444",needle:{start:"rgba(240,    128,    128,    1)",end:"rgba(255,    160,    122,
.9)"}},highlights:[{from:20,to:10000,color:"#eee"}]};var
y=0,E=this,n=0,I=0,H=!1;this.setValue=

function(a){n=b.animation?y:a;var                                          d=(b.maxValue-
b.minValue)/100;I=a>b.maxValue?b.maxValue+d:a<b.minValue?b.minValue-
d:a;y=a;b.animation?k():this.draw();return
this};this.setRawValue=function(a){n=y=a;this.draw();return
this};this.clear=function(){y=n=I=this.config.minValue;this.draw();return
this};this.getValue=function(){return
y};this.onready=function(){};l(this.config,b);this.config.minValue=parseFloat(this.config.m
inValue);this.config.maxValue=parseFloat(this.config.maxValue);b=this.config;n=

y=b.minValue;if(!b.renderTo)throw Error("Canvas element was not specified when creating the
Gauge                                                                    object!");var
z=b.renderTo.tagName?b.renderTo:document.getElementById(b.renderTo),a=z.getContext("2d"),A,
C,D,t,u,f,B;q();this.updateConfig=function(a){l(this.config,a);q();this.draw();return
this};var              M={linear:function(a){return              a},quad:function(a){return
Math.pow(a,2)},quint:function(a){return      Math.pow(a,5)},cycle:function(a){return      1-
Math.sin(Math.acos(a))},bounce:function(a){a:{a=1-a;for(var b=0,

c=1;;b+=c,c/=2)if(a>=(7-4*b)/11){a=-Math.pow((11-6*b-11*a)/4,2)+Math.pow(c,2);break
a}a=void       0}return       1-a},elastic:function(a){a=1-a;return       1-Math.pow(2,10*(a-
1))*Math.cos(30*Math.PI/3*a)}},G=null;a.lineCap="round";this.draw=function(){if(!A.i8d){B.c
```

```
learRect(-t,-u,C,D);B.save();var
g={ctx:a};a=B;p();N();J();d();s();b.title&&(a.save(),a.font=24*(f/200)+"px
Arial",a.fillStyle=b.colors.title,a.textAlign="center",a.fillText(b.title,0,-
f/4.25),a.restore());b.units&&(a.save(),a.font=22*(f/200)+"px Arial",

a.fillStyle=b.colors.units,a.textAlign="center",a.fillText(b.units,0,f/3.25),a.restore());A
.i8d=!0;a=g.ctx;delete              g.ctx}a.clearRect(-t,-u,C,D);a.save();a.drawImage(A,-t,-
u,C,D);if(Gauge.initialized)L(),K(),H||(E.onready&&E.onready(),H=!0);else                var
e=setInterval(function(){Gauge.initialized&&(clearInterval(e),L(),K(),H||(E.onready&&E.onre
ady(),H=!0))},10);return this}};Gauge.initialized=!1;

(function(){var                          b=document,l=b.getElementsByTagName("head")[0],q=-
1!=navigator.userAgent.toLocaleLowerCase().indexOf("msie"),v="@font-face       {font-family:
'Led';src:                                                    url('fonts/digital-7-
mono."+(q?"eot":"ttf")+"');}",k=b.createElement("style");k.type="text/css";if(q)l.appendChi
ld(k),l=k.styleSheet,l.cssText=v;else{try{k.appendChild(b.createTextNode(v))}catch(e){k.css
Text=v}l.appendChild(k);l=k.styleSheet?k.styleSheet:k.sheet||b.styleSheets[b.styleSheets.le
ngth-1]}var g=setInterval(function(){if(b.body){clearInterval(g);

var
e=b.createElement("div");e.style.fontFamily="Led";e.style.position="absolute";e.style.heigh
t=e.style.width=0;e.style.overflow="hidden";e.innerHTML=".";b.body.appendChild(e);setTimeou
t(function(){Gauge.initialized=!0;e.parentNode.removeChild(e)},250)}},1)})();Gauge.Collecti
on=[];

Gauge.Collection.get=function(b){if("string"==typeof                              b)for(var
l=0,q=this.length;l<q;l++){if((this[l].config.renderTo.tagName?this[l].config.renderTo:docu
ment.getElementById(this[l].config.renderTo)).getAttribute("id")==b)return      this[l]}else
return"number"==typeof                                        b?this[b]:null};function
domReady(b){window.addEventListener?window.addEventListener("DOMContentLoaded",b,!1):window
.attachEvent("onload",b)}

domReady(function(){function                                              b(b){for(var
e=b[0],d=1,g=b.length;d<g;d++)e+=b[d].substr(0,1).toUpperCase()+b[d].substr(1,b[d].length-
1);return                                                                 e}for(var
l=document.getElementsByTagName("canvas"),q=0,v=l.length;q<v;q++)if("canv-
gauge"==l[q].getAttribute("data-type")){var
k=l[q],e={},g,p=parseInt(k.getAttribute("width"),10),w=parseInt(k.getAttribute("height"),10
);e.renderTo=k;p&&(e.width=p);w&&(e.height=w);p=0;for(w=k.attributes.length;p<w;p++)if(g=k.
attributes.item(p).nodeName,"data-type"!=g&&"data-"==

g.substr(0,5)){var                          d=g.substr(5,g.length-5).toLowerCase().split("-
");if(g=k.getAttribute(g))switch(d[0]){case
"colors":d[1]&&(e.colors||(e.colors={}),"needle"==d[1]?(d=g.split(/\s+/),e.colors.needle=d[
0]&&d[1]?{start:d[0],end:d[1]}:g):(d.shift(),e.colors[b(d)]=g));break;case
"highlights":e.highlights||(e.highlights=[]);g=g.match(/(?:(?:-?\d*\.)?(-?\d+){1,2}
){2}(?:(?:#|0x)?(?:[0-9A-F|a-f]){3,8}|rgba?\(.*?\))/g);for(var   d=0,J=g.length;d<J;d++){var
s=g[d].replace(/^\s+|\s+$/g,"").split(/\s+/),x={};

s[0]&&""!=s[0]&&(x.from=s[0]);s[1]&&""!=s[1]&&(x.to=s[1]);s[2]&&""!=s[2]&&(x.color=s[2]);e.
highlights.push(x)}break;case
"animation":d[1]&&(e.animation||(e.animation={}),"fn"==d[1]&&/^\s*function\s*\(/.test(g)&&(
g=eval("("+g+")")),e.animation[d[1]]=g);break;default:d=b(d);if("onready"==d)continue;if("m
ajorTicks"==d)g=g.split(/\s+/);else    if("strokeTicks"==d||"glow"==d)g="true"==g?!0:!1;else
```

```
if("valueFormat"==d)if(g=g.split("."),2==g.length)g={"int":parseInt(g[0],10),dec:parseInt(g
[1],10)};else continue;

e[d]=g}}e=new                                                    Gauge(e);k.getAttribute("data-
value")&&e.setRawValue(parseFloat(k.getAttribute("data-value")));k.getAttribute("data-
onready")&&(e.onready=function(){eval(this.config.renderTo.getAttribute("data-
onready"))});e.draw()}});window.Gauge=Gauge;

<!-- Gauge Code Ends -->

</script>

<script type="text/javascript">

 var voltaje;

 var intensidad;

 var potencia;

 var frecuencia;

 var fpot;

 var eact;

function update() {   <!--esta funcion es la que me interesa --> recoge los datos del
arduino -->

 var xhttp;

  xhttp = new XMLHttpRequest();

  xhttp.onreadystatechange = function() {

    if (xhttp.readyState == 4 && xhttp.status == 200) {

          var json = JSON.parse(xhttp.responseText);

          voltaje = json.voltaje;

          intensidad = json.intensidad;

          potencia = json.potencia;

          frecuencia = json.frecuencia;

          fpot = json.fpot;

          eact = json.eact;

    }

        };
```

```
  xhttp.open("GET", "http://192.168.1.50/ajax", true); <!-- la ip por donde se mandan los
datos y la peticion de datos -->

  xhttp.send();

   }

 var id_interval = setInterval('update()',5000);

</script>



</head>

<body>

<header>

<h1><CENTER><IMG
SRC="https://www.upc.edu/++theme++homeupc/assets/images/logomark.png"></IMG></CENTER></h1>

<H2><CENTER>Arduino Monitor</H2></CENTER><br />

<H3><CENTER>Monitorea datos instalaci&oacuten fotovoltaica</CENTER></H3>

</header>

<br />

<main><CENTER>

<p> VOLTAJE [V]</p> <p> INTENSIDAD [A]</p> <p> POTENCIA [W]</p><br />

<p><canvas id="an_gauge_1" data-major-ticks="0 40 80 120 160 200 240 280 320 360" data-
type="canv-gauge"   data-min-value="0"   data-max-value="340"   data-onready="setInterval(
function() { Gauge.Collection.get('an_gauge_1').setValue(voltaje);}, 200);"></canvas></p>

<p><canvas id="an_gauge_2" data-major-ticks="0 2 4 6 8 10 12 14 16 18 20" data-type="canv-
gauge"  data-min-value="0"  data-max-value="20"  data-onready="setInterval(  function()  {
Gauge.Collection.get('an_gauge_2').setValue(intensidad);}, 200);" ></canvas></p>

<p><canvas id="an_gauge_3" data-major-ticks="0 500 1000 1500 2000 2500 3000 3500 4000 4500
5000"     data-type="canv-gauge"     data-min-value="0"     data-max-value="5000"     data-
onready="setInterval( function() { Gauge.Collection.get('an_gauge_3').setValue(potencia);},
200);" ></canvas></p>

<p> FRECUENCIA [Hz]</p> <p> FACTOR DE POTENCIA </p> <p> ENERGIA ACTIVA [W.h]</p><br />

<p><canvas id="an_gauge_4" data-major-ticks="0 10 20 30 40 50 60 70 80 90 100" data-
type="canv-gauge"   data-min-value="0"   data-max-value="100"   data-onready="setInterval(
function()     {     Gauge.Collection.get('an_gauge_4').setValue(frecuencia);},     200);"
></canvas></p>
```

```
<p><canvas id="an_gauge_5" data-major-ticks="0 0.10 0.20 0.30 0.40 0.50 0.60 0.70 0.80 0.90
1" data-type="canv-gauge" data-min-value="0" data-max-value="1" data-onready="setInterval(
function() { Gauge.Collection.get('an_gauge_5').setValue(fpot);}, 200);" ></canvas></p>

<p><canvas id="an_gauge_6" data-major-ticks="0 1000 2000 3000 4000 5000 6000 7000 8000 9000
10000"    data-type="canv-gauge"    data-min-value="0"    data-max-value="10000"    data-
onready="setInterval(  function()  {  Gauge.Collection.get('an_gauge_6').setValue(eact);},
200);" ></canvas></p>

<br /><br /><br /><br /><footer> <b><br /><CENTER>

Juan Carlos Ruiz - Trabajo final de Grado - Ing. Electr&oacutenica industrial y
autom&aacutetica </CENTER><br /></b> <footer>

</body>

</html>
```

6. Pruebas

Una vez implementada la instalación y el software se procede a realizar las pruebas funcionales del proyecto. Para ello se conectará el analizador de red entre la instalación de red y una carga durante un periodo largo de tiempo con la intención de recoger los datos generados por el consumo de la misma.

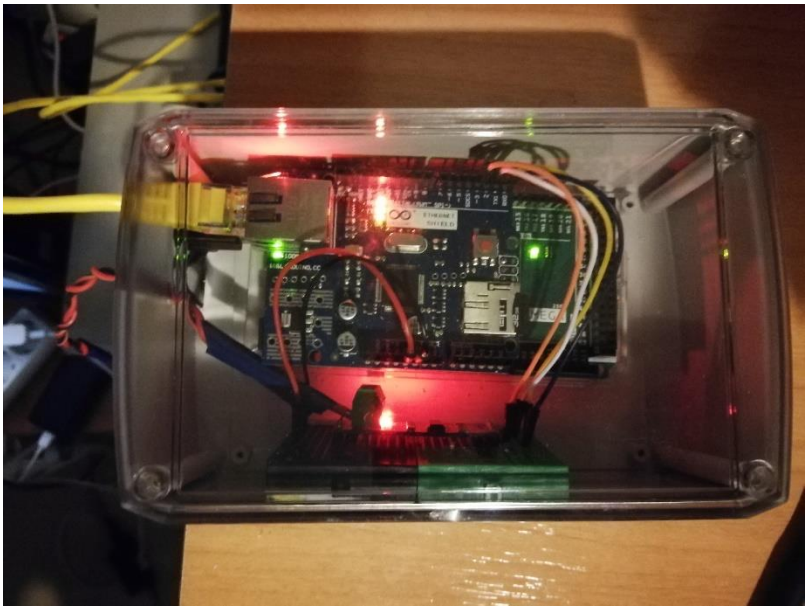


Figura 6.1. Dispositivo en funcionamiento (Fuente: Juan Carlos Ruiz).

A su vez, se comprobará que el histórico se genera correctamente en la tarjeta SD de la placa ARDUINO Ethernet Shield y que se envían los datos en tiempo real correctamente a través de la red, consultando la dirección IP que se ha configurado la comunicación. A continuación se presentan las pruebas realizadas junto con los resultados obtenidos.

Se ha conectado el dispositivo entre las 19:48 y las 19:54 horas de la tarde, se han conectado y desconectados diferentes cargas durante este periodo simulando así el uso normal de una instalación en la que los diferentes dispositivos conectados a la red van variando en el tiempo. El registro grabado en la tarjeta SD es el siguiente:

id;timestamp;v_med(V);i_med(A);pactiva_med(W);frec_med(Hz);fpot_med;eact_med(Wh);
1;19:48;218.52;4.72;1029.18;50.00;0.65;5616.40;
2;19:50;218.30;4.72;1027.28;50.00;0.65;5655.12;
3;19:52;217.77;4.70;1021.50;50.00;0.65;5693.85;
4;19:54;218.29;4.72;1027.00;50.00;0.65;5732.41;

Tal como se observa, se han grabado 4 registros de datos durante el periodo de 6 minutos aproximadamente. El primer registro grabado es el de cabecera, donde se indican los datos y sus unidades de medida grabados a continuación.

Durante este periodo se han consultado los datos online a través de la página web:

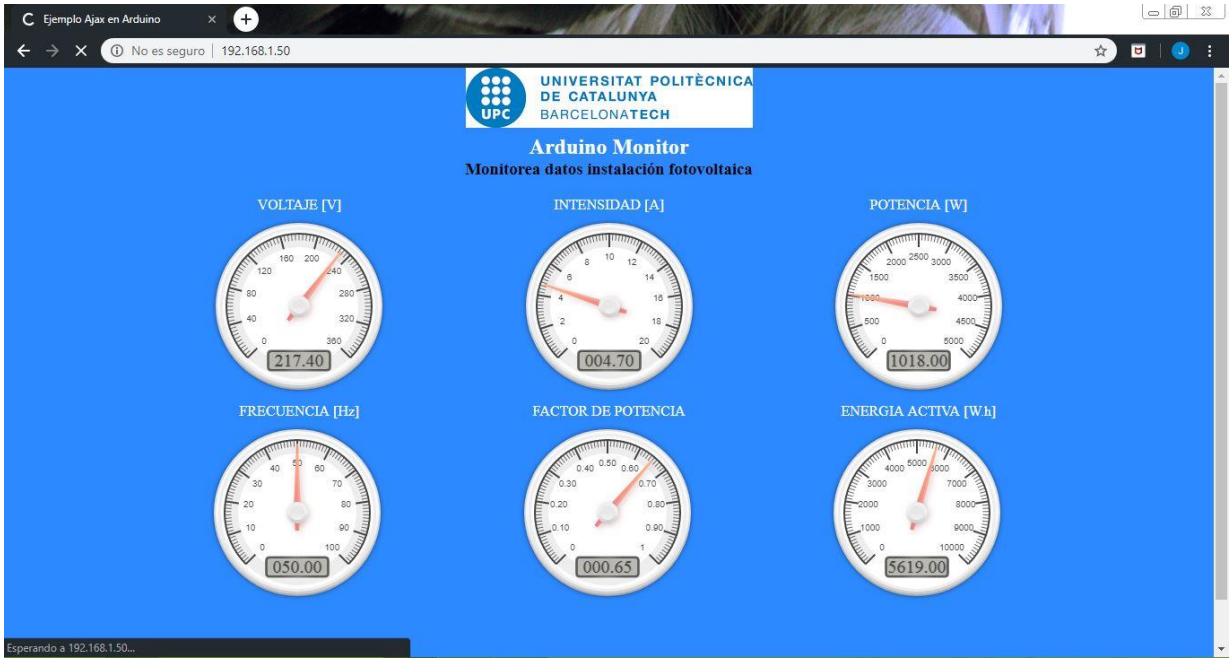


Figura 6.2. Web en funcionamiento (Fuente: Juan Carlos Ruiz).

Una vez comprobado el correcto funcionamiento de la instalación, se exportan los datos a Excel para evaluarlos.

id	timestamp	v_med(V)	i_med(A)	pactiva_med(W)	frec_med(Hz)	fpot_med	eact_med(Wh)
1	19:48	218,52	4,72	1029,18	50	0,65	5616,4
2	19:50	218,3	4,72	1027,28	50	0,65	5655,12
3	19:52	217,77	4,7	1021,5	50	0,65	5693,85
4	19:54	218,29	4,72	1027	50	0,65	5732,41

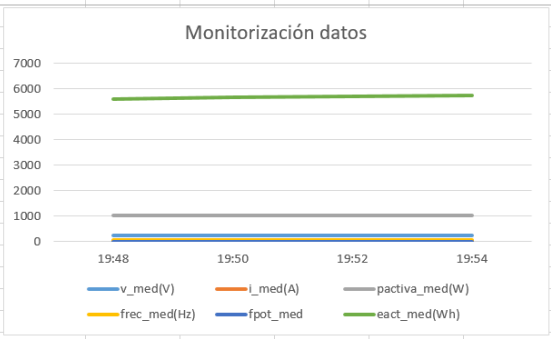


Figura 6.3. Datos exportados en Excel (Fuente: Juan Carlos Ruiz).

Tal como se observa en la **Figura 6.3** los datos se exportan correctamente con el formato esperado, pudiendo así establecerse patrones de comportamiento de la instalación para posibilitar posteriormente aumentar la eficiencia de la misma.

7. Análisis del impacto ambiental

En este apartado se pretende cuantificar el impacto medioambiental de nuestro proyecto tanto a nivel de consumo y de huella ecológica como de riesgos de su implementación.

A parte del consumo energético eléctrico se ha calculado el gasto energético de los componentes que se deberán implementar en la instalación fotovoltaica.

Según su documento técnico el analizador de red tiene un consumo de 3VA.

Calculando la energía que esto supone durante todo un año funcionando las 24 horas del día obtenemos que el consumo anual del dispositivo analizador de red es 26.28 kWh.

El consumo de la placa Arduino MEGA con la shield ethernet es de aproximadamente 50mA, lo que a lo largo del año suponen 2.19 kWh.

El consumo total anual de la instalación será por lo tanto de 28.47 kWh, o lo que es lo mismo 18.5 kg de CO2.

Teniendo en cuenta que el dispositivo estará bien aislado se le calcula una vida útil de unos 10 años aproximadamente, con lo cual la huella ecológica durante todo este tiempo será de 185 kg de CO2.

Los riesgos que podrían derivar en un consumo más elevado del esperado no serán otros que los ambientales. Un mal aislamiento de la placa en condiciones de lluvia o humedad podría hacer que el consumo del prototipo se elevara o incluso dejará de funcionar, debiendo en este último caso sustituirse por una nueva placa, elevando costes y recursos para su implementación.

Conclusiones

La realización de este proyecto ha servido para comprender todas las fases del proceso de prototipado de una aplicación demandada por un cliente o usuario.

Teniendo que comprender la funcionalidad de lo demandado para diseñar su resolución de la manera más eficiente comparando las diferentes posibilidades.

Las mayores dificultades que nos hemos encontrado durante la realización del mismo han sido sobre todo la necesidad de obtener conocimientos en ámbitos y lenguajes de programación no trabajados con anterioridad.

Pese a que durante el grado se ha trabajado en lenguaje C, se han tenido que definir y acotar las posibilidades de la plataforma Arduino.

Además se ha tenido que hacer hincapié en adquirir conocimientos básicos en HTML y JavaScript para poder relizar la comunciación entre Arduino y el entorno gráfico del proyecto.

Tras iniciar el proyecto sobre la placa ARDUINO UNO se dedujo que la capacidad de memoria de la placa seleccionada reducía considerablemente el alcance del proyecto. De tal manera que nos vimos obligados a cambiar la placa del prototipo por una ARDUINO MEGA 2560 con mucha mas capacidad de memoria y que nos ha permitido implementar la funcionalidad completa del proyecto.

Con las actuales características del proyecto, se conseguirá mostrar de forma correcta y comprensible el trabajo realizado por la instalación fotovoltaica a la que se conecte y recopilar datos interesantes para el análisis de la eficiencia de la misma.

De cara al futuro, existen diferentes posibilidades para ampliar el presente proyecto que podrian ser interesantes.

Dado que las comunicaciones entre los diferentes dispositivos ya esta implementada, las mejoras vendrian en cuanto al apartado más gráfico, quizás dotando al proyecto de una página web mejorada y/o implantando la misma dentro de la web de la universidad.

También ampliando los datos obtenidos desde el analizador o mejorando el tratamiento y análisis sobre ellos.

Presupuesto y/o Análisis Económico

Un punto importante dentro de la realización de un proyecto de estas características es el de valorar económicamente el prototipo, y ver si sería viable y económicamente posible su comercialización en un futuro.

Tan importante como el correcto funcionamiento del producto, es que el mismo sea rentable para la empresa que lo comercializa.

A continuación, se desglosarán los diferentes costes generados de la construcción del prototipo.

Tal como se verá a continuación, no se han contabilizado los costes de las licencias de los distintos softwares utilizados para la implementación del programa del ya que en este caso son de libre uso.

Costes materiales

A continuación, se detalla el coste de todos los componentes y material que se ha utilizado en la implementación proyecto.

	Cantidad	PVP (€)	Coste total (€)
Placa Arduino MEGA	1	39	39
Placa Ethernet Shield	1	13	13
Placa RS485 Shield	1	3	3
Transformador 230 a 9V 1A	1	9,52	9,52
Caja RETEX serie 102 155x95x60	1	7,37	7,37
Otros elementos	1	4,77	4,77
			Coste total (€)
TOTAL			76,66

Costes de ingeniería

En este apartado, se calculan los costes que supondría el trabajo de ingeniería realizado en el proyecto en el caso que una empresa pagará a un alumno en prácticas de la UPC.

El precio propuesto por la UPC por hora de un estudiante en prácticas es de un máximo de 20 € / h. En esta valoración se ha impuesto un precio de 10 € / h.

	Horas	Precio/hora (€)	Coste total (€)
Tiempo dedicado al estudio de la viabilidad del proyecto	38	10	380
Tiempo dedicado al diseño de hardware	20	10	200
Tiempo dedicado al diseño del software	100	10	1000
Tiempo dedicado al montaje del hardware	4	10	40
Tiempo dedicado a la depuración del software	15	10	150
Tiempo dedicado a pruebas funcionales	5	10	50
Tiempo dedicado a la confección de la documentación	25	10	250
	PVP (€)	IVA 21% (€)	Coste total (€)
TOTAL	2070	434,7	2504,7

Hipótesis del valor de mercado

A continuación, se hará una hipótesis sobre el coste de comercialización y los beneficios que se podrían obtener en el caso de que se implementara en cadena.

En la tabla siguiente se muestran los costes de los componentes utilizados teniendo en cuenta los descuentos que realizan los diferentes proveedores a clientes que realizan pedidos importantes de material.

De esta manera se elaborará un presupuesto lo más parecido posible al de comercialización.

	Cantidad	PVP (€)	Descuento (%)	Coste total (€)
Placa Arduino UNO	1	39	25	29,25
Placa Ethernet Shield	1	13	25	9,75
Placa RS485 Shield	1	3	25	2,25
Transformador 230 a 9V 1A	1	9,52	25	7,14
Caja RETEX serie 102	1	7,37	25	5,53
Otros elementos	1	4,77	25	3,58
				Coste total (€)
TOTAL				57,50

En este apartado, se muestran los costes derivados del trabajo de un ingeniero y un técnico para realizar el proyecto.

El precio por hora de cada uno de ellos es el valor real orientativo que una empresa paga a estos profesionales.

El tiempo también es orientativo de lo que puede tardar un profesional del sector experimentado y con las herramientas adecuadas para la realización del trabajo.

	Horas	Precio/hora (€)	Coste total (€)
Tiempo dedicado al estudio de la viabilidad del proyecto	5	56	280
Tiempo dedicado al diseño de hardware	7	56	392
Tiempo dedicado al diseño del software	40	56	2240
Tiempo dedicado a la depuración del software	5	56	280
Tiempo dedicado a la confección de la documentación	15	56	840
	PVP (€)	IVA 21% (€)	Coste total (€)
TOTAL	4032	846,72	4878,72

	Horas	Precio/hora (€)	Coste total (€)
Tiempo dedicado al montaje del hardware	3	38	114
Tiempo dedicado a pruebas funcionales	1	38	38
	PVP (€)	IVA 21% (€)	Coste total (€)
TOTAL	152	31,92	183,92

A continuación, se hace un estudio que relaciona los diferentes gastos derivados de la construcción del prototipo con el precio de venta, para de esta manera estudiar el beneficio que generaría a la empresa la construcción y venta del producto.

Se supone un sueldo de 1700 euros mensuales para el ingeniero y 1200 euros para el técnico.

Suponiendo 40/h de trabajo semanales y un mes de 4 semanas:

	Horas	Precio/hora (€)	Coste total (€)
Compra de material			57,50
Gasto trabajo ingeniero	72	10,5	756
Gasto trabajo técnico	4	7,5	30
Venta del producto			XXXX
GASTO NETO			-843,50

Del estudio se concreta que para que la comercialización del producto sea rentable, el precio de venta deberá ser superior a 830 euros por unidad.

Bibliografía

Aguado, O. (2018). *Principales diferencias entre Raspberry Pi y Arduino*. [online] Conmasfuturo. Available at: http://www.conmasfuturo.es/principales-diferencias-entre-raspberry-pi-y-arduino-2/ [Accessed 19 Feb. 2018].

Anon, (2018). [online] Available at: http://www.leantec.es/blog/24_Como-conectar-Arduino-a-Ethernet-y-hacer-serv.html [Accessed 19 Feb. 2018].

Aprendiendo Arduino. (2018). *Diferentes modelos de placas Arduino*. [online] Available at: https://aprendiendoarduino.wordpress.com/2015/03/23/diferentes-modelos-de-placas-arduino/ [Accessed 19 Feb. 2018].

Aprendiendo Arduino. (2018). *Memoria Flash, SRAM y EEPROM*. [online] Available at: https://aprendiendoarduino.wordpress.com/2017/06/21/memoria-flash-sram-y-eeprom-3/ [Accessed 19 Feb. 2018].

Aprendiendoarduino.wordpress.com. (2018). *RS485 | Aprendiendo Arduino*. [online] Available at: https://aprendiendoarduino.wordpress.com/category/rs485/ [Accessed 19 Feb. 2018].

Arduino.cc. (2018). *Arduino - Home*. [online] Available at: https://www.arduino.cc/ [Accessed 19 Feb. 2018].

Doutel, F. (2018). *Raspberry Pi frente a Arduino: ¿quién se adapta mejor a mi proyecto maker?*. [online] Xataka.com. Available at: https://www.xataka.com/makers/raspberry-pi-frente-a-arduino-quien-se-adapta-mejor-a-mi-proyecto-maker [Accessed 19 Feb. 2018].

Ingeniería y Tecnología (Arduino-Processing). (2018). *Ingeniería y Tecnología (Arduino-Processing)*. [online] Available at: http://jasonhd.wixsite.com/jason/proyecto-joas-1 [Accessed 19 Feb. 2018].

Naylampmechatronics.com. (2018). *Comunicación RS485 con Arduino*. [online] Available at: http://www.naylampmechatronics.com/blog/37_Comunicaci%C3%B3n-RS485-con-Arduino.html [Accessed 19 Feb. 2018].

Prometec.net. (2018). *Ethernet SHIELD W5100 R3 | Tienda y Tutoriales Arduino*. [online] Available at: https://www.prometec.net/producto/ethernet-shield-w5100-r3/ [Accessed 19 Feb. 2018].

Prometec.net. (2018). *MAX 485 TTL to RS485 | Tienda y Tutoriales Arduino*. [online] Available at: https://www.prometec.net/producto/max-485-ttl-to-rs485/ [Accessed 19 Feb. 2018].

Rinconingenieril.es. (2018). *Citar un sitio web - Cite This For Me*. [online] Available at: https://www.rinconingenieril.es/rs485/ [Accessed 19 Feb. 2018].

WooCommerce, C. (2018). *Arduino Ethernet (Original) » IBEROBOTICS*. [online] IBEROBOTICS. Available at: https://www.iberobotics.com/producto/arduino-ethernet-original/?gclid=Cj0KCQjwsNfOBRCWARIsAGITapaVDPzqmSE5vP-TZFfJRSZ6LheKT0wRpZ1EX_pyWRlSgdbxEfXbPCkaAriZEALw_wcB [Accessed 19 Feb. 2018].

WooCommerce, C. (2018). *Arduino UNO Rev3 (Original) » IBEROBOTICS*. [online] IBEROBOTICS. Available at: https://www.iberobotics.com/producto/arduino-uno-rev3-original/ [Accessed 19 Feb. 2018].

YouTube. (2018). *Conectar Arduino al televisor*. [online] Available at: https://www.youtube.com/watch?v=ny6fzcoz454 [Accessed 19 Feb. 2018].

Ethernet SHIELD W5100 R3 | Tienda y Tutoriales Arduino, 2018. *Prometec.net* [online],

Ingeniería y Tecnología (Arduino-Processing), 2018. *Ingeniería y Tecnología (Arduino-Processing)*[online],

MAX 485 TTL to RS485 | Tienda y Tutoriales Arduino, 2018. *Prometec.net* [online],

Memoria Flash, SRAM y EEPROM, 2018. *Aprendiendo Arduino* [online],

RS485 | Aprendiendo Arduino, 2018. *Aprendiendoarduino.wordpress.com* [online],

WOOCOMMERCE, CREADO, 2018, Arduino Ethernet (Original) » IBEROBOTICS. *IBEROBOTICS*[online]. 2018. [Accessed 19 February 2018]. Available from: https://www.iberobotics.com/producto/arduino-ethernet-original/?gclid=Cj0KCQjwsNfOBRCWARIsAGITapaVDPzqmSE5vP-TZFfJRSZ6LheKT0wRpZ1EX_pyWRlSgdbxEfXbPCkaAriZEALw_wcB

WOOCOMMERCE, CREADO, 2018, Arduino UNO Rev3 (Original) » IBEROBOTICS. *IBEROBOTICS* [online]. 2018. [Accessed 19 February 2018]. Available from: https://www.iberobotics.com/producto/arduino-uno-rev3-original/

Arduino - Home, 2018. *Arduino.cc* [online],

Citar un sitio web - Cite This For Me, 2018. *Rinconingenieril.es* [online],

Comunicación RS485 con Arduino, 2018. *Naylampmechatronics.com* [online],

Ethernet SHIELD W5100 R3 | Tienda y Tutoriales Arduino, 2018. *Prometec.net* [online],

Fritzing, 2018. *Fritzing.org* [online],

Ingeniería y Tecnología (Arduino-Processing), 2018. *Ingeniería y Tecnología (Arduino-Processing)*[online],

MAX 485 TTL to RS485 | Tienda y Tutoriales Arduino, 2018. *Prometec.net* [online],

Memoria Flash, SRAM y EEPROM, 2018. *Aprendiendo Arduino* [online],

RS485 | Aprendiendo Arduino, 2018. *Aprendiendoarduino.wordpress.com* [online],

(C) 2018, W.A. SMITH, 2018, Arduino SD Card Web Server using Ethernet Shield. *Startingelectronics.org* [online]. 2018. [Accessed 20 February 2018]. Available from: https://startingelectronics.org/tutorials/arduino/ethernet-shield-web-server-tutorial/SD-card-web-server/

¿Qué es AJAX? | Digital Learning, 2018. *Digitallearning.es* [online],

Arduino - Ethernet, 2018. *Arduino.cc* [online],

Display Analógico con JavaScript | Tienda y Tutoriales Arduino, 2018. *Prometec.net* [online],

Luis Llamas, 2018. *Luis Llamas* [online],

Manual HTML en español, 2018. *Uv.es* [online],

MEDINA, PATRICIO, 2018, Aprovechar al máximo la memoria de tu Arduino - MCI Capacitación. *MCI Capacitación* [online]. 2018. [Accessed 20 February 2018]. Available from: http://cursos.mcielectronics.cl/como-aprovechar-al-maximo-la-memoria-de-tu-arduino/

RUIZ, ANTONIO, RUIZ, ANTONIO and PERFIL, VER, 2018, Arduino en español. *Manueldelgadocrespo.blogspot.com.es* [online]. 2018. [Accessed 20 February 2018]. Available from: http://manueldelgadocrespo.blogspot.com.es

SOY?, ¿QUIÉN, 2018, Convertir un número a cadena en Arduino con print, sprintf y dtostrf. *Programar fácil con Arduino* [online]. 2018. [Accessed 20 February 2018]. Available from: https://programarfacil.com/blog/arduino-blog/conversion-de-numeros-a-cadenas-en-arduino/

Tutorial HTML - Tabla de contenidosspatutorial - HTML.net, 2018. *Es.html.net* [online],

ARDUINO PROJECTS BOOK, 2017. ,

Prácticas con Arduino 2, 2018. , EDUBÁSICA.

Anexo A: Materiales

A1. Cuadro de materiales

Material	Cantidad	PVP (€)	Coste total (€)
ARDUINO MEGA	1	39	39
ETHERNET SHIELD	1	13	13
RS485 SHIELD	1	3	3
Transformador universal 230V a 3-12V 1A	1	9.52	9.52
Caja RETEX serie 102 125x75x50	1	7.37	7.37
Otros componentes	1	4.77	4.77
TOTAL			**76.66**

Anexo B: Información técnica

A2. ARDUINO MEGA

1 Arduino Mega2560

Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

The Mega 2560 is an update to the Arduino Mega, which it replaces.

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-9V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB (8 KB used by bootloader)
SRAM	8 KB
EEPROM	4 KB (ATmega328)
Clock Speed	16 MHz

A3. Ethernet Shield W5100

Overview

W5100 Ethernet shield is a WIZnet W5100 breakout board with POE and Micro-SD designed for Arduino platform. 5V/3.3V compatible operation voltage level makes it compatible with Arduino boards, leafmaple, and other Arduino compatible board.

Features

- With Micro SD interface
- 5V/3.3V double operational voltage level
- 10Mb/100Mb Ethernet socket with POE
- All electronic brick interface are broken out
- Operation temperature: -40℃ ~ +85℃

Specifications

PCB size	55.88mm X 68.58mm X 1.6mm
Indicators	TX,RX,COL,FEX,SPD,LNK
Power supply	5V
Communication Protocol	SPI
RoHS	Yes

Electrical Characteristics

Specification	Min	Type	Max	Unit
Power Voltage	3V	-	5.5	VDC
Input Voltage VH:	3	-	5.5	V
Input Voltage VL:	-0.3	0	0.5	V
Current	-	-	100	mA

Figure 1 Top Map

Arduino PIN	Description
D0	Rx/Breakout
D1	TX/Breakout
D2	Breakout
D3	Breakout
D4	SD_CS
D5	Breakout
D6	Breakout
D7	Breakout
D8	Breakout
D9	W5100_Reset
D10	W5100_CS
D11	MOSI
D12	MISO
D13	SCK

A0	Breakout
A1	Breakout
A2	Breakout
A3	Breakout
A4	IIC_SDA/Breakout
A5	IIC_SCL/Breakout

A4. Analizador de red CVM-SP-RS485-C

1.1.- Parameters measured and calculated by CVM-SP.

By means of an internal microprocessor, following parameters can be simultaneously monitored:

Parameter	*Symbol*	*Parameter*	*Symbol*
Voltage	*V*	Power factor	*PF*
Current	*A*	Frequency	*Hz*
Active power	*kW*	Maximum demand	*Pd*
Reactive power	*kvarL /(-C)*	kW. h	*energy*
Voltage THD	*% THD- V*	kvarh. L	*energy*
Current THD	*% THD- A*	kvarh. C	*energy*

1.2.- CVM-SP types.

Type	**Currents**	**Hardware Setting**	**Code**
CVM-SP	25A	0232	770 480
	100A	0230	770 484
CVM-SP-RS485-C (Communications and output relay)	25A	1332	770 481
	100A	1330	770 485

All CVM-SP types do measurements over two quadrants: **only facilities consuming energy.**

2.1.- Connection arrangement

Before powering the instrument for the first time, verify following points:

a.- **Power supply/measuring voltage:** see lable on the instrument side.

- ***Standard power supply***	:	*Single-phase **230 V** ~ (a.c.)*
- *Frequency*	:	*50 - 60 Hz*
- *Power supply tolerance*	:	*+ 15 / - 10 %*
- *Instrument burden*	:	*3 VA*

b.- Maximum admissible current: see lable on the instrument side

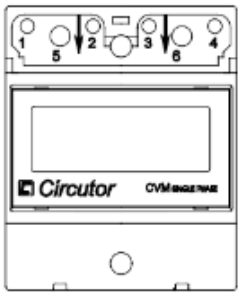
c.- Operation conditions:

- Working temperature range:		-10 ºC to +50 ºC
- Relative humidity	:	5 a 95 % RH (non-condensing)
- Height	:	until 2000 m

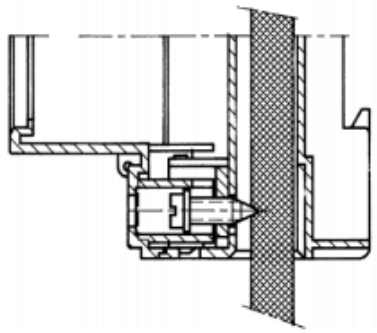
d.- Safety:

- Protection against electric shock by class II double-isolation
- Designed for class III facilities - 300 V a.c (EN 61010).

2.2.- Connection terminal arrangement

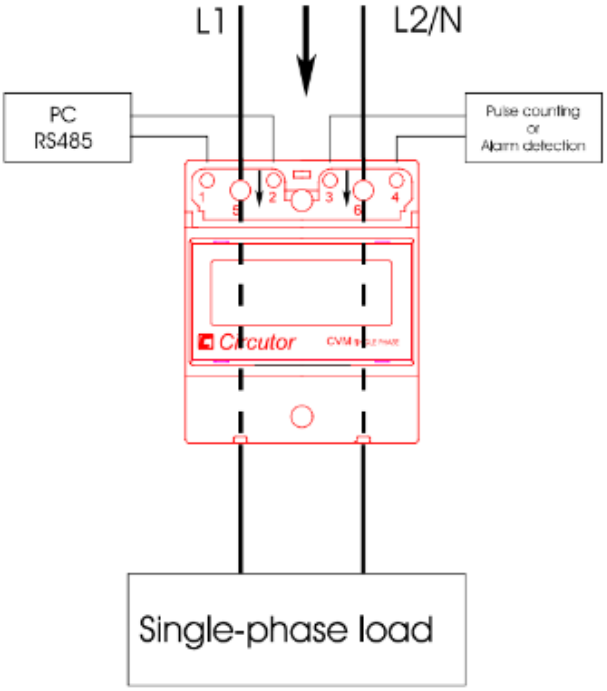


Terminal No.	Terminal description	
1	*RS-485 (-)*	(Acc. to type)
2	*RS-485 (+)*	(Acc. to type)
3	*Relay RL1 output*	(Acc. to type)
4	*Relay common*	(Acc. to type)
5	*Voltage/Current L1 input*	
6	*Voltage N/L2 input*	



DETAIL OF THE VOLTAGE MEASURING MODE

2.3.- Connection diagram of a CVM-SP to a single-phase power system:



A5. Caja RETEX serie 102 155x95x60

Serie 102

Caja para aplicaciones en que se requiera una tapa transparente, por ejemplo equipos con LCDs o contadores que requieran inspección visual. Tapa en plástico policarbonato de alta resistencia color bronce fumé. Indicada también para colocación de teclados de membrana de hasta 1.2 mm.

Moderna estética con aristas biseladas y arcos laterales. Cierre por tornillos M3, que es posible ocultar con tapones autoadaptables, incluidos.
Doble área de rebaje en tapa y base, que permite la fijación de teclados de membrana.
Acabado pulido que permite una fácil limpieza de la superficie.
Máxima capacidad interior.
Torretas para montaje de circuitos en todas las bases.

Fabricación: base en plástico ABS resistente hasta temperaturas de 85°C. Tapa en policarbonato de alta resistencia mecánica. Soporta temperaturas de hasta 120°. Tornillería de montaje M3. Tapones en PVC espumado.

Color estándar: similar a gris claro RAL 7035.

Unidad de embalaje: tapa, base, 4 tornillos y 4 tapones.

Opciones:
- mecanizaciones especiales.
- serigrafía.
- pintado interno para blindaje contra radiofrecuencia EMI / RFI, con pintura de base grafítica, cobre o níquel dependiendo del nivel de protección requerido.
- ABS autoextinguible, según grado UL 94 V0.
- otros colores.

(Otras opciones consulte a su distribuidor).

Caixa para aplicações que necessitem de uma tampa transparente, por exemplo, equipamentos com LCDs ou contadores que precisam de inspeção visual. Tampa em plástico policarbonato de alta resistência, cor bronze fumado. Indicada também para colocação de teclados de membrana de até 1,2 mm.

Estética moderna com arestas biseladas e arcos laterais. Fecho por parafusos M3, que é possível ocultar com tampões auto-adaptáveis, incluídos.
Dupla área de entalhe na tampa e base, que permite a fixação de teclados de membrana.
Acabamento polido que permite uma limpeza fácil da superfície.
Máxima capacidade interior.
Torretas para montagem de circuitos em todas as bases.

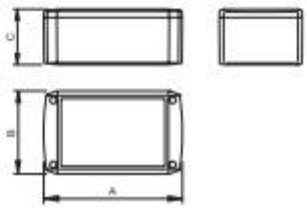
Fabricação: *base em plástico ABS resistente à temperaturas de até 85°C. Tampa em policarbonato de alta resistência mecânica. Suporta temperaturas de até 120°C. Parafusos de montagem M3. Tampões em PVC espumado.*

Cor standard: *similar ao cinza claro RAL 7035.*

Unidade de embalagem: *tampa, base, 4 parafusos e 4 tampões.*

Opções:
- *mecanizações especiais.*
- *serigrafia.*
- *pintura interna para blindagem contra radiofrequência EMI / RFI, com tinta de base grafítica, cobre o níquel dependendo do nível de protecção necessário.*
- *ABS autoextinguível, de acordo com o grau UL 94 V0.*
- *outras cores.*

(Outras opções consulte o seu distribuidor).



+

CODE	Color Cor	A	B	C
33102001		90	50	35
33102002		110	60	40
33102003	Gris claro RAL 7035	125	75	50
33102004	Cinza claro RAL 7035	155	95	60
33102005		190	115	75
33102006		220	140	90

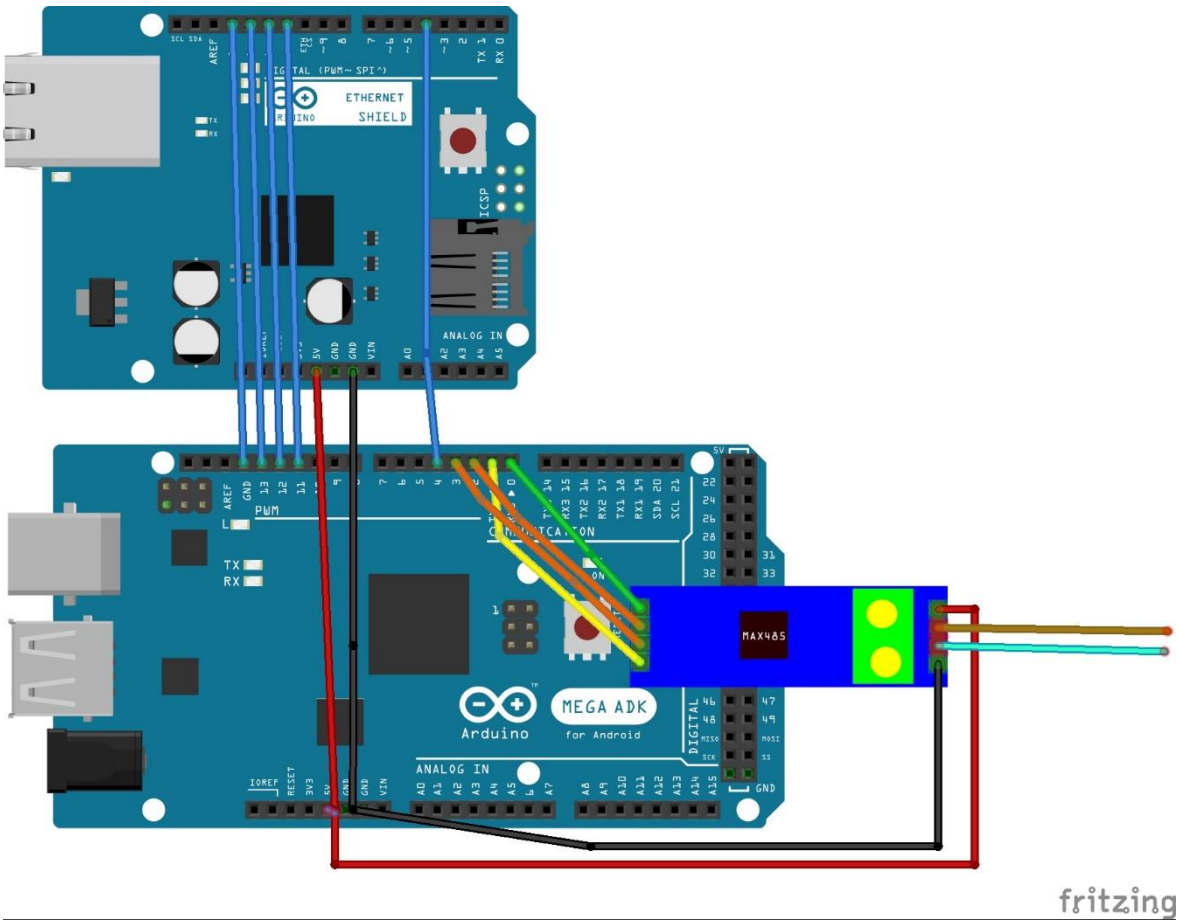
A6. Transformador universal 230 a 3..9V 1A

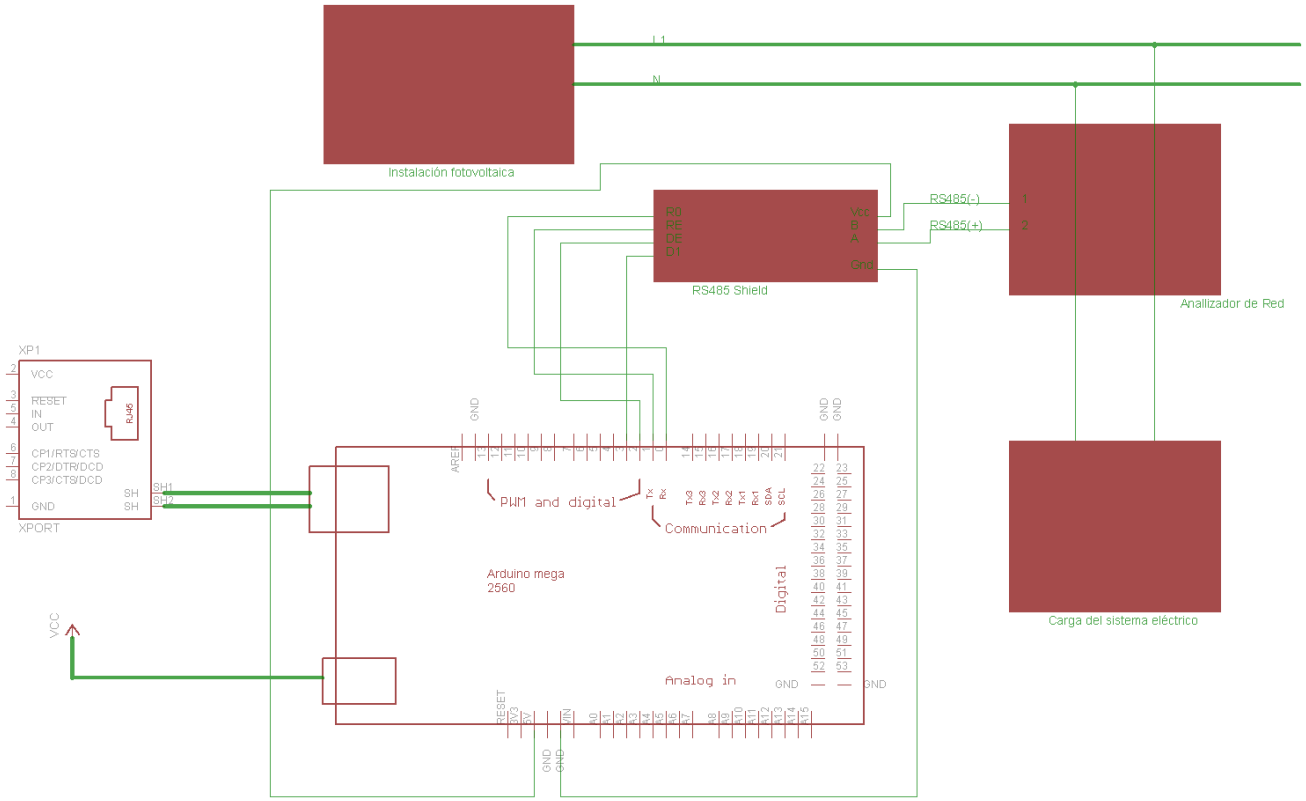
DESCRIPCIÓN

Caracteristicas:

Tipo de alimentador	adaptador de alimentación, por impulso, universal
Campo de tensiones de alimentación	100...240 V CA
Parametros eléctricos de salida	3V/4.5V/5V/6V/7.5V/9V/12V/1000mA,
Corriente de salida máx.	1000 mA
Conector de salida	kit de enchufes universales
Medidas interiores	82 x 35 x 50 mm
Peso	128 g

Anexo C: Esquemas Electrónicos




Instalación fotovoltaica
L1
N
RS485(-)
RS485(+)
1
2
Analizador de Red
Carga del sistema eléctrico
RS485 Shield
R0
RE
DE
D1
Vcc
B
A
Gnd
XP1
VCC
RESET
IN
OUT
CP1/RTS/CTS
CP2/DTR/DCD
CP3/CTS/DCD
GND
SH
SH
RJ45
XPORT
VCC
Arduino mega
2560
PWM and digital
Communication
Digital
Analog in
GND
AREF